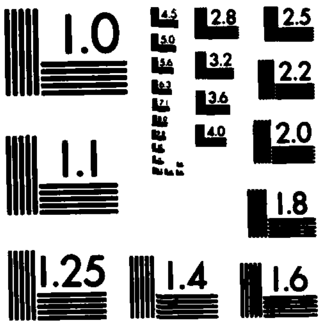
AD-A144 271 C3I (COMMAND CONTROL COMMUNICATIONS AND INTELLIGENCE) 1/2
INFORMATION SYSTEMS..(U) INSTITUTE FOR DEFENSE ANALYSES
ALEXANDRIA VA T C BARTEE ET AL. APR 84 IDA-P-1774
UNCLASSIFIED IDA/HQ-84-28529 MDA903-79-C-0018 F/G 9/2 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1

AD-E500 675

Copy 8 of 50 copies

12

AD-A144 271

IDA PAPER P-1774

C^{3}I INFORMATION SYSTEMS INTERNETWORK STUDY

T. C. Bartee
O. P. Buneman
J. M. McQuillan
S. T. Walker

April 1984

DTIC FILE COPY

Prepared for
Assistant Secretary of Defense
(Communications, Command, Control and Intelligence)

IDA

INSTITUTE FOR DEFENSE ANALYSES

84 08 13 001

IDA Log No. HQ 84-28529

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A144271	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) C^3I Information Systems Internetwork Study		5. TYPE OF REPORT & PERIOD COVERED FINAL October 1982 - December 1983
		6. PERFORMING ORG. REPORT NUMBER IDA PAPER P-1774
7. AUTHOR(s) T.C. Bartee, O.P. Buneman, J.M. McQuillan, S.T. Walker		8. CONTRACT OR GRANT NUMBER(s) MDA 903 79 C 0018
9. PERFORMING ORGANIZATION NAME AND ADDRESS Institute for Defense Analyses 1801 N. Beauregard Street Alexandria, VA 22311		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Task T-3-163
11. CONTROLLING OFFICE NAME AND ADDRESS Director, Information Systems ASD(C^3I), The Pentagon Washington, D.C. 20301		12. REPORT DATE April 1984
		13. NUMBER OF PAGES 111
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) DoD-IDA Management Office 1801 N. Beauregard Street Alexandria, VA 22311		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE —

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

None

18. SUPPLEMENTARY NOTES

N/A

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Data Base Management Systems, protocols, local area networks, security, link encryption, end-to-end encryption, Ada, transport level protocols, logistics

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report has four subjects:

(1) An investigation of DoD protocol standards for computer communications. emphasis is on the X2.5 protocol and a similar existing DDN protocol.

(2) Local area network protocols are discussed with emphasis on security architecture.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. (Continued)

(3) A study of the logistics usage of AUTODIN and an analysis of a way to reduce that usage.

(4) A study of data base management systems and their use in data communications systems. Interfaces to Ada are emphasized in this section.

IDA PAPER P-1774

C^{3}I INFORMATION SYSTEMS INTERNETWORK STUDY

T. C. Bartee
O. P. Buneman
J. M. McQuillan
S. T. Walker

April 1984

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street
Alexandria, Virginia 22311

Contract MDA 903 79 C 0018
Task T-3-163

CONTENTS

ABBREVIATIONS

AFLC	Air Force Logistics Command
ASD	Assistant Secretary of Defense
AUTODIN	Automatic Digital Network
BIU	Basic Interface Unit
BBN	Bolt, Beranek and Newman
CCITT	Consultative Committee--International Telegraph and Telephone
C^3I	Communications, Command, Control and Intelligence
CONUS	Continental United States
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
DAAS	Defense Automated Accounting System
DARPA	Defense Advanced Research Projects Agency
DCA	Defense Communications Agency
DCE	Data Connection Equipment
DDN	Defense Data Network
DES	Data Encryption System
DIA	Defense Intelligence Agency
DLA	Defense Logistics Agency
DLANET	Defense Logistics Agency Teleprocessing Network
DoD	Department of Defense
DoDIIS	Department of Defense Information Systems Internet Study
DTE	Data Terminal Equipment
FTP	File Transfer Protocol
HDLC	High-Level Data Link Control
HFE	Host Front End
IP	Internetwork Protocol
IPLI	Internet Private Line Interface
IPMS	Interpersonal Messaging Service
I-SA AMPE	Inter-Service/Agency Automatic Message Processing
ISO	International Standards Organization
JCS	Joint Chiefs of Staff
LAN	Local Area Network

NVT	Network Virtual Terminal
PAD	Packet Assembler/Disassembler
PBX	Private Branch Exchange
PLI	Private Line Interface
RFNM	Ready for Next Message
SDC	System Development Corporation
SDLC	Synchronous Data Link Control
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol
WIS	WWMCCS Intercomputer Network

I. INTRODUCTION

A. BACKGROUND MATERIAL

In October 1982, the Institute for Defense Analyses was requested by the Director, Information Systems, ASD(C^3I), to study Local Area Networks and, in particular, the security aspects of these networks. Since many DoD local area networks will be connected into larger networks, in particular the DDN, this aspect of their usage also was to be studied. Important relevant subjects in this general area included the protocols now being developed by DoD and the user interfaces having to do with data bases and allied programming and query languages.

There were two additional related areas to be studied: (1) the logistics usage of AUTODIN with emphasis on reducing the load on that network, and (2) a comparison of X.25 and the 1822 protocol with a view toward possible adoption of X.25 by DDN.

B. OVERVIEW OF STUDY AREAS

The security architectures for local area networks utilize two general approaches which can be used in either pure or mixed forms in a given network. The first of these is to form a secure network using physical security. This can be realized in many ways, but generally includes placing the transmission media--coax, for example--in hardened pipe and then keeping the pipe visible for inspection. There are several allied techniques and generally the network areas are also kept physically secure to some security level using conventional means, and transmitters and receivers are placed in "safe" enclosures. The second general technique involves encrypting the data. The encryption is often end-to-end in local area networks where the header is in the clear because other receivers, and often the network itself, must have header information to perform properly. Link encryption is then used as required.

The DoD first developed a device called a Private Line Interface (PLI), which will encrypt data while having the header clear, and a new improved device called an Internet Private Line Interface (IPLI) is now in development. IPLIs are moderately expensive, however, and the keys required to update these devices must be distributed and updated at intervals, and this can be a problem in system operation.

Considerations concerning the above can be found in later sections on security in this report, as can considerations concerning the partitioning of users into communities of interest and blocking the passage of inappropriate security level data into (or from) receivers (and transmitters).

The Defense Data Network (DDN) is now progressing in its development, and the operation of this network appears to be satisfactory from most viewpoints and ahead of schedule in many cases. As a result, the demand for usage of this network now exceeds original estimates and still more user communities appear to be suitable candidates for possible inclusion into the network. In many cases the potential users make heavy application of commercial hardware and software which has not been substantially modified for DoD usage. This provides cost savings and interoperability within prescribed equipment circles (IBM-to-IBM; DEC-to-DEC, for example) but can be a problem when widespread interoperability is considered. Fortunately, several International Standards exist or are in preparation which show promise of making manufacturer-to-manufacturer interoperability possible. The U.S. Government and DoD often participate heavily in development of these standards, and development and usage by DoD plays an important role in the success of many of these protocols.

A particular protocol called X.25 is now widely used commercially, and many logistics (and other) users have expressed a desire to use X.25 as an access level protocol to the DDN which now uses a protocol called 1822 (the number was the number of a Bolt, Beranek and Newman report describing the protocol, see Section II). A study of this protocol and instructions on its usage, along with a comparison of X.25 and 1822, can be found in Section II.

The logistics community was the original developer of the AUTODIN network. As time passed, however, many other users joined this network and it has grown and prospered in recent years. At present, however, there are plans to phase out AUTODIN by the early 1990s. At the same time, usage of the network continues to accelerate, largely because present users find their needs increasing.

Because logistics users are still the major users of the AUTODIN network, a study was made to determine if some of the load could be removed without disrupting service and without economic penalty. This appeared to be judicious, not just for normal operation but also because the network plays a vital role in plans for crisis situations, and at this time there would reasonably be an increase in usage of the logistics community which might affect many planned operations. Our study of this subject shows that a substantial decrease in the logistics load can be made in a reasonable, cost-effective way, and this is reported on in this paper. Section IV deals with this in more detail, giving many statistics on present usage.

C. CONCLUSIONS AND RECOMMENDATIONS

The studies which were made have yielded several general conclusions and recommendations, which follow. Supporting evidence and the reasoning behind these conclusions and recommendations are in Sections II through V.

1. The X.25 protocol is a good protocol for DDN when properly used, and the DoD specification now being worked on appears satisfactory. There are no features in 1822, which is now used in DDN, which are not also in X.25. Care must be used in specific cases, and a list of precautions is in Section II. There is no planned interface between the IPLI in development and X.25, so IPLI users must use 1822. Further, X.25 and 1822 cannot interface directly, so command and control and other users requiring encryption of data must still use 1822. Despite this limitation, we recommend adoption of X.25 for the DDN for unclassified users.
2. The logistics community can reduce the present load on AUTODIN by about 20 percent by unloading a major part of the usage by the two DAAS sites. A description of how this can be accomplished can be found in Section IV. The Defense Logistics Agency (DLA) is moving on this subject and has already cancelled its recently issued request for bids for new AUTODIN I processors. Further, there is a movement, including meetings and studies, toward consideration of the DAAS traffic in conjunction with the DLA network, with a view toward using DDN as a primary carrier. We feel this is an excellent direction for study and future action.
3. The widespread use of Ada® which is now planned has considerable implication concerning data base management systems in DoD C^3I networks. The query languages to be used and interfaces are just now being worked on, and some considerations and suggestions in this area can be found in Section V.
4. The techniques to achieve local area network security are only now in development. The best prospect for

Ada® is a registered trademark of the Ada Joint Program Office (AJPO) of the U.S. Department of Defense.

near-term use appears to involve a combination of physical security for communications media and "safes" for transmitters and receivers, along with some software or hardware at transmitters and receivers to block data transfer with unsatisfactory security levels or compartmental designations. More technical details of this strategy can be found in Section III. More study and work in this area needs to be done before a completely satisfactory architecture, with the necessary software and hardware, is arrived at, however, and key distribution continues to be a major problem.

5. There are three high-level protocols now proposed for DoD usage which show considerable promise. They are (1) MIL-STD-1788 File Transfer Protocol, (2) MIL-STD-1782 Telenet Protocol, and (3) MIL-STD-1781 Simple Mail Transfer Protocol. These are three useable and well-thought-out protocols for DoD purposes. Our chief reservation concerns the Mail Transfer Protocol because new IBM protocols show promise of becoming industry standards and this fact needs to be considered in future plans.

II. A COMPARISON OF DDN INTERFACE STANDARDS

A. INTRODUCTION

This chapter discusses protocols for the Defense Data Network (DDN). It specifically compares two access standards, the "DDN" X.25 and 1822, which are protocols for the interconnection of host computers to DDN. Then, some implications of the choice of access protocol on other DDN design decisions are presented. This is followed by some information on higher-level protocols, in particular the mail protocols.

1. Terminology

Before beginning analysis, it will be useful to review the seven-level reference model developed by the International Standards Organization (Ref. 1). Known formally as the Basic Reference Model for Open Systems Interconnection, the ISO model has become the commonly accepted way of analyzing and understanding network functions. The names and numbers of the levels of the model will be used below. From the lowest, or most basic level, they are defined as follows:

(1) The physical interface, or connection (e.g., RS-232).

(2) The data link protocol. For example, HDLC, SDLC.

(3) The network level. Responsible for routing between users and for control of the flow of data.

(4) The transport level. Responsible for user-to-user error recovery.

(5) The session level. Sets up and breaks connections on behalf of the user.

(6) The presentation level. Provides for formatting of data so that both user and network can operate on the form most efficient for their purpose.

(7) The application level. The individual user selects the application level protocols in ISO model. This is the level where data base systems and application programs reside.

2. <u>The Basic Question</u>

The two interface specifications in question, X.25 and 1822 (described in more detail below), are standards for protocol levels (1) through (3). It is possible, and perhaps even desirable, for these two approaches to coexist on DDN. It may also be possible and desirable to use both of these lower-level specifications to support the same higher-level protocols when interoperability is required.

The main questions to be addressed here are: How do these two specifications compare? Are there inherent advantages or disadvantages of one or the other for C^3I users of DDN? Are there any cautions or guidelines to be followed? (A list of "watch out's" is provided below.) What are the implications of this choice in other areas, especially in higher protocol levels?

B. THE 1822 STANDARD

1. <u>Background</u>

The access protocol used in previous DoD packet switching networks is commonly referred to as "1822", since its specification is contained in BBN Report No. 1822, originally published in 1969 (Ref. 2).

To understand the origins and evolution of 1822, it is helpful to recall that at the time of the design of the ARPANET there were no standards for data communications protocols as we know them. The ISO model had not been developed. X.25 had not yet been designed. HDLC had not been formalized. Most data communications were accomplished with vendor-specific protocols such as IBM's BSC. The designers of the ARPANET wanted to make an interface specification that would be more general, that would permit connection of computers of different sizes and speeds.

The original 1822 interface solved this problem with a one-of-a-kind connection, composed of several signals in parallel. The signalling provided for the transfer of bits in both directions simultaneously, at a mutually determined rate, according to a "hand shake" technique. It also provided conventions for determining whether the other end of the connection was ready to send or receive data. This early interface design, a set of protocols at levels 1 and 2, is now obsolescent. The same functionality is available at level 1 with RS 232 and MIL STD 188, and at level 2 with HDLC and HDH (Ref. 2). For the purposes of this analysis, it does not matter whether a DDN subscriber uses the old-style parallel signalling of 1822, or the newer HDH. Both cases will be designated as "1822".

The significant differences between 1822 and X.25 come at level 3. This level specifies the use of the 1822 ARPANET Leader, a 96-bit descriptor assigned to each message passed between a host and the network. In the host-to-network direction, the leader of a data message carries the number of the destination network, IMP, and host. It also defines the type of message (datagram or virtual circuit), its length (0 to 8063 bits), and an identifying number. Control messages can also be sent from the host to the network to indicate various kinds of trouble, which are separately identified. Messages flowing from the network to the host have similar leaders, with one important additional message type, the RFNM (Ready for Next Message), which tells the host that the specified message reached its destination.

These facilities are similar, but not identical, to those offered by most level 3 protocols. In particular, it would be possible to translate this protocol into the X.25 level 3, but not the reverse, since 1822 has a different and broader range of features.

2. <u>Usage</u>

The use of the 1822 interface specification has been confined to the ARPANET and similar networks, almost all supplied by BBN to the DoD community. It is a good interface, which has stood the test of time, but it has not stimulated any significant commercial spinoffs. BBN has tried, with little success, to sell commercial networks with the 1822 interface. Most commercial customers demand X.25. A few small companies have made a business from supplying 1822 hardware and software for various kinds of computers, but these companies remain an insignificant force in the larger data communications marketplace. 1822 represents an evolutionary dead end. It was once leading-edge technology, but events have overtaken it.

3. <u>Implementation Questions--"Watch Out's"</u>

The 1822 interface is so well-understood within the community of interest that there are relatively few cautions that need to be mentioned. The most important are:

- Implementation investment. Most computer types are already connected to ARPANET. The technology should be borrowed, copied, or otherwise acquired rather than reinvented. This will minimize both cost and risk. It is generally not a good investment of time and effort to develop new 1822 hardware and software for types of computers not yet connected to DDN.
- Quality of service. It is important within 1822 to specify the nature of the network service desired. This includes the choice between datagrams and virtual circuits, as well as the handling of error conditions and blocking.
- Efficiency of host software. Experience has shown that network performance is usually limited by the speed of

the software in the host computers, including the 1822 handler, and higher protocols such as TCP, etc. Performance tuning usually pays for itself in this area.

4. Decision Factors

A prospective DDN user considering the use of 1822 for access to DDN should consider the following factors:

- Is it necessary or desirable to have datagram service? Members of the ARPANET community working on packet voice have found that datagrams are preferable to virtual circuits for their application. DDN X.25 does not offer datagram service. For this reason, we would expect that DARPA will continue to support the 1822 standard for certain users.
- Is it important to protect an investment in 1822 technology on a particular computer? For some computer systems, 1822 was introduced years ago with considerable difficulty and expense. Some of these systems are approaching the end of their useful lives. In such cases, it would be unwise to invest in a new implementation of X.25, even at low cost. In other cases, the maintenance of 1822 technology for several computers may be the responsibility of a single user community. It may be cheaper and simpler to stay with 1822 for all of those computers rather than converting them to X.25.

C. THE X.25 STANDARD

1. Background

CCITT Recommendation X.25 is an international standard for the connection of computers and terminals to data networks (Ref. 3). X.25 defines the way in which a packet device, called in the standard a "packet DTE" (Data Terminal Equipment), attaches to a network port or "DCE" (Data Connection Equipment). It defines the physical device interface (level 1), the link protocol, a form of High-Level Data Link Control (HDLC), and a unique protocol to combine many user conversations onto a single data link (level 3).

The DDN X.25 specification is compliant with FIPS Publication 100 and Federal Standard 1041, which are previously established for the use of X.25 with the Federal Government and within the Department of Defense (Ref. 4). It is very important to note that this specification is <u>not</u> exactly the same as the CCITT Recommendation X.25 adopted by most carriers.

CCITT Recommendation X.25 contains many options and implementation choices. FIPS 100/Federal Standard 1041, which specifies the general use of X.25 for the Federal Government, defines some of the choices left open in X.25. In areas where X.25 allows a choice, a single choice appropriate for DDN is specified in Ref. 4; in areas which X.25 leaves unspecified, addressing in particular, conventions are specified that are consistent with the overall architecture of DDN. The intent of this approach is to make DDN service available to hosts in a way that requires no changes to a host DTE implementation that is compliant with FIPS 100/Federal Standard 1041 and CCITT Recommendation X.25. If hosts implement the extensions described in Ref. 4, users will be able to take advantage of additional DDN features required in military networks, such as precedence and logical addressing.

The DDN X.25 DCE supports all facilities specified as E (essential) by FIPS 100/Federal Standard 1041, and most facilities specified as A (additional). The additional facilities <u>not</u> supported are:

(1) datagrams and associated facilities, and

(2) bilateral closed user groups.

A final consideration is how to link X.25 users with other X.25 users, who may be using different computer systems, and with non-X.25 users. As stated in Ref. 4:

> A key goal of the DDN X.25 implementation is interoperability among all DDN subscribers. That is, effective

communication should be possible, not only between subscribers attached to the DDN using identical vendor-supplied X.25 implementations, but between subscribers using different X.25 implementations, and between a subscriber using an X.25 interface to the DDN and a subscriber using an 1822 interface to the DDN.

The DDN X.25 DCE offers <u>two</u> types of service to X.25 DTEs:

(1) DDN Standard X.25 Service, which, when used in conjunction with DoD standard protocols, provides interoperable communication between an X.25 DTE and other DDN hosts that also implement the DoD standard protocols, whether they are connected to DDN via the 1822 interface or via the X.25 interface;

(2) DDN Basic X.25 Service, which provides communication only between an X.25 DTE and other DDN X.25 DTEs implementing compatible higher-level protocols.

This is a very critical aspect of the DDN X.25 interface, one that will be analyzed in more detail below.

2. <u>Usage</u>

The primary use of X.25 is to connect a computer to a public packet network such as Telenet or Tymnet. These networks usually charge for service based on a flat fee per port plus a usage charge. Because X.25 provides a user with a protocol supporting multiple data conversation on a single link, a computer connection through an X.25 port can support the same number of logical channels as several PAD (packet assembler/disassembler) ports into the network, but at a lower cost. The single X.25 link usually costs less for the communication interface hardware on the user's computer.

X.25 is also used in private packet networks such as DDN as a concentration protocol either to save on the cost of computer ports or to permit connection of more terminals than a computer can physically attach.

Some form of X.25 is currently supported by the following organizations (this list is not exhaustive):

Amdahl
Associated Computer Consultants
Atlantic Research
AT&T
BBN
Burroughs
Cableshare
Codex
Com-Pro
Control Data Corporation
Dartmouth Time Sharing
Data General
Datapoint
DEC
Dynatech Packet Technology
Gandalf
GTE Telenet
Harris
Hewlett-Packard
Honeywell
IBM Corporation
ICOT
Infotron
I.P. Sharp
Memotec
National Semiconductor
NCR/Comten
Northern Telecom
Pacific Software Manufacturing
Perkin-Elmer
Prime
SESA/Honeywell
Siemens
Spectron
Stratus
Systar
Tandem
Texas Instruments
Tymnet
Univac
Varian Associates

There are over 20 countries with X.25 public data networks. In addition, there are at least five manufacturers each of facsimile systems, local area networks, and digital PBXs who support X.25 (Refs. 5, 6, 7).

3. Implementation Questions--"Watch Out's"

Not all X.25 implementations offer the same level of user support and functionality. Even though the protocol is standardized, variations in the features available can conflict with the requirements of other offerings. Prospective C^3I users of DDN X.25 should evaluate the following issues:

- Number of logical channels which may be assigned on a link. Too small a number is an artificial limit on the number of terminals and users that can be supported.
- Maximum speed of the line. Attaching to a network at a higher speed will provide improved performance levels or allow more users for the same level of performance.
- Support of the X.29 PAD control protocol. Special packet types are used to control a terminal attached via a PAD. Host support for X.29 will simplify user programming.

- Support for optional facilities. X.25 contains a number of special features--closed user groups, datagram service, etc.--which are not supported by DDN. In addition, some of the X.25 features which <u>are</u> supported in DDN may <u>not</u> be supported by all computer vendors.
- Operating system and application support. Not all operating systems available for a computer system may support X.25. Specialized applications packages, such as data base query languages and various embedded or specialized DoD applications, may not support users connected via X.25.
- Memory and other hardware requirements. Some computers will require hardware upgrades or expansions.
- Address length. DDN may interconnect with several of the U.S. and overseas public packet networks. If so, users should be aware of the international standards permitting addresses up to 15 digits in length. Users connecting through DDN to an international packet network need to have host software which can accommodate as many address digits as the called network and international routing conventions require.
- Level of vendor support. Any X.25 implementation should support all nonoptional functions of level 2 and level 3. DoD users should beware of vendor statements such as the product is "based on" or "similar to" or "supports at level 2," used to qualify the support. If the full standard is not implemented, it can have serious implications for interoperability.
- Level of DDN support, and matching vendor features with DDN. To quote from Ref. 8:

  > If DDN provides a sophisticated high-performance X.25 interface (optimized for TCP hosts for example), then an "off-the-shelf" X.25 host software package will probably <u>not</u>

take advantage of most DDN X.25 features. It may in fact not be able to operate without some enhancement. DDN should not be forced to provide a service which is the lowest common denominator of services provided by commercial carriers. Since hardly any broadly applicable X.25 hosts exist, and any X.25 host will probably need to undergo enhancements, we will strive to provide good DDN X.25 service unconstrained by current host implementations.

- The implications for host planners are clear: do not assume that the existing vendor-supplied and supported X.25 package will work well (or at all) in DDN without modification.
- Basic versus Standard X.25 service. One of the most important implementation questions facing the DDN is whether to enforce connection to DDN using a Standard X.25, with the DoD standard protocol architecture of TCP, IP, FTP, and SMTP, or whether to allow use of the vendor-supplied protocols for computer systems (e.g., IBM's SNA or equivalent). The former choice offers universal interoperability with any other computers which support the DoD standard. The latter offers a lower cost solution to partial interoperability with similar computer systems. As noted in the section on internetworking below, users of basic X.25 service will be limited in their ability to communicate with users on other networks and with classified users.

4. Decision Factors

A user deciding on the use of X.25 for access to DDN should consider the following questions:

- Will there be a need to connect the same hosts and terminals to public packet networks like Telenet and Tymnet? Some users look upon such public networks as their backup. If so, X.25 is advantageous.
- Is it desirable to use an access method widely available on different host computers? X.25 is an international

standard, so it is supported by most of the major computer companies and can be used to link computers of different speeds, types, and manufactures.

- Will terminals be connected by X.25 as well as host computers? They will need to be connected to PADs, at extra cost.
- Do the application programs to send and receive data over DDN exist already, or will they have to be written? It is more expensive to modify existing telecommunications applications written for an earlier generation network than to develop new software with X.25 in mind.
- Do the computers and terminals support X.25 already? Have they been installed already? New equipment can be purchased for X.25 compatibility, reducing or eliminating conversion costs. However, much of the anticipated usage of DDN will come from older equipment which does not support X.25 today.
- If the X.25 support is in place already, is it compatible with DDN? As noted above, this issue may be the cause of some unpleasant surprises. The X.25 package must be compatible with FIPS 100/Federal Standard 1041, and validated for use on DDN. Furthermore, it may require enhancement to take advantage of DDN features.
- Does the user organization have personnel with data communications experience, especially with X.25? Software work on X.25 can be difficult without experienced staff. Users without this kind of experience in-house may wish to look closely at vendor support for network hardware and software.

D. KEY ISSUES

DDN is a large-scale system which at present has more than one computer network. DDN's goals include:

- quality of service
- low cost
- security
- low risk
- interoperability

Each of these points is discussed briefly below:

1. Quality of Service

It is of paramount importance that DDN service remain highly reliable and survivable at all times. Of secondary weight, but still of major importance is the quality of service offered to users, measured in delay and throughput. Interface specifications such as X.25 and 1822 contribute directly to these goals since DDN is only as good as the access lines into it.

2. Cost

There are many cost elements to be considered in selecting an access method for DDN:

- cost of acquisition/development
- cost/efficiency of operation
- cost of maintenance

The hardware and software to operate X.25 and 1822 represent a relatively small part of the overall DDN budget, but a significant amount of money in themselves, when hundreds of host computers are involved. Clearly, it is the total of all cost elements over the life cycle of use of DDN that is of most interest.

3. Security

There are no particular security issues with regard to the choice between X.25 and 1822 for this network. With regard to classified but sensitive data DES encryption might be used when required.

Use of the IPLI, which is an end-to-end encryption device is another matter, however. Here, the use of X.25 raises important design questions, which are treated at length in Ref. 4. At the time of this writing, the IPLI X.25 architecture has not yet been specified. This means C^3I users are almost forced to 1822.

4. Risk

The risk of failure in DDN should be kept to extremely low levels, considering the national security implications of a major problem. There are several categories of risk to be considered, most especially:

- implementation risk
- operation risk

The choice of interface standard is not a major risk element in DDN. Nevertheless, if one interface offers a lower risk of failure, it may be an important consideration in the decision.

5. Interoperability

There are at present two groups of DDN subscribers--those host computers which run vendor or other special protocols, and those which use the standard DoD protocol architecture, including TCP/IP. These two groups cannot communicate with each other, even if they both use the same interface standard for levels 1 through 3, either X.25 or 1822.

On the other hand, the choice of interface standard for a particular host computer dictates which other computers it will interoperate with. Choosing basic X.25 means that the system can communicate only with compatible X.25 DTEs (the same holds true for selecting 1822 and vendor protocols at higher levels). Choosing standard X.25 (or 1822 plus TCP/IP) means that communication is possible with any computers that implement TCP/IP, regardless of interface standard. Thus, interoperability depends almost entirely on the choice of the higher protocols.

E. COMPARISON AND EVALUATION

1. Similarities

(1) The multiple channel capability of X.25 allows users to exchange data between several tasks at the same

time, and still to support terminals on one system accessing programs or data on another. 1822 offers the same capability.

(2) X.25 is full-duplex and can support simultaneous transmission in both directions, reducing interference between users. The same is true of 1822.

(3) The maximum message length is similar--8192 bits for X.25 and 8063 bits for 1822. For DDN standard (interoperable) service, even this difference is irrelevant, since the limiting factor is the maximum IP packet, which is 576 octets (4608 bits). These lengths are adequate for good host computer efficiency.

(4) Both protocols use the same levels 1 and 2 (considering only the high-level data version of 1822). Furthermore, both protocols can support the same protocols at levels 4 through 7.

2. Differences

The overriding difference between the two specifications for unclassified users is that X.25 is a very successful national, Federal, and international standard. This has benefits for C^3I users in the form of lower implementation costs, more flexibility, better availability of compatible equipment, and a larger pool of technical talent from which to draw.

3. Evaluation

Issue	X.25	1822
Quality of service	(Equal)	
Cost		
Operation	(Equal)	
Acquisition	Low	High
Maintenance	Low	High
Security	(To be determined)	
Risk	Low	Medium
Interoperability	(Depends on higher levels)	

F. IMPLICATIONS

A real issue in DDN protocol selection is vendor protocols versus DoD protocols.

1. Identification of Protocol Architecture

X.25 DTEs employing the DoD standard TCP/IP protocol architecture must indicate this by means of the call user data field of the CALL REQUEST packet. It is important to note (Ref. 4) that:

> Indication of the use of the DoD standard protocol architecture is independent of the selection of DDN standard or DDN basic X.25 service ... Therefore, a host employing the DoD standard protocol architecture and using DDN standard X.25 service must include both the DDN standard X.25 service facility and the call user data DoD standard protocol identification in its CALL REQUEST packet.

X.25 users of non-DoD protocol architectures may use any identification recognized by the other end.

2. Internetworking and Security

As explained in Ref. 4, the choice between TCP/IP and vendor protocols has significant implications. Choosing not to implement TCP/IP will limit the ability of C^3I users to communicate with users on other networks and to gain classified network access.

> The Defense Data Network is an Internetwork environment. That is, DDN as a whole is made up of a number of constituent packet-switching networks that are interconnected via gateways. Communication across gateways requires the use of the Internet Protocol, which, for a host accessing DDN using X.25, requires that the host implement the DoD standard protocol architecture and employ DDN standard X.25 service. In addition, a classified host is attached to a DDN constituent network of lower classification by means of an Internet Private Line Interface (IPLI). IPLIs, which themselves contain gateways, also require the use of the Internet Protocol; moreover, they do not, as currently designed, offer an X.25 host interface. These attributes of the DDN Internet have two implications for users of DDN basic X.25 service:

(1) DDN hosts that do not implement IP and higher-level DDN protocols, and which use only DDN basic X.25 service, cannot communicate across gateways. Their network communication is therefore restricted to a single DDN constituent network.

(2) X.25 hosts cannot be provided classified service on a constituent network of lower classification. Should X.25 host access be developed for the IPLI in the future, classified network access will be made available to hosts using DDN standard X.25 service only.

3. Interoperability

There are two levels of interoperability that might be considered valuable for users. The first is available with DDN basic X.25 service and vendor protocol architecture for levels 4 through 7. This permits users with the same protocol architecture to exchange information in many cases. This is the main requirement, since all of the computers in a particular application area may be of the same type (e.g., HIS 6000, IBM 370, DEC VAX, etc.).

The second level, or true interoperability, requires implementation of the DoD standard protocols TCP and IP, as well as the higher-level protocols which implement DDN standard services, when such services are provided by the host: the Telnet Protocol for character-oriented terminal support, the File Transfer Protocol (FTP) for file movement between hosts, and the Simple Mail Transfer Protocol (SMTP) for communication between electronic mail service hosts.

4. Higher-Level Protocols--Electronic Mail

As a common and important example of a higher-level protocol, consider electronic mail systems that operate "on top of" either X.25 or 1822. The proposed standard for this application is Simple Mail Transfer Protocol (SMTP). SMTP has been developed to take advantage of TCP and IP functions. It is independent of the particular transmission subsystem(s), requiring only a reliable ordered data stream, such as is provided by

TCP. Further, SMTP is capable of relaying mail across one or more networks. This means two hosts can exchange mail across an internetwork environment, such as the DDN. It even provides for electronic mail to be relayed from the source computer to the destination through an intermediary computer system. Thus, mail can be relayed between hosts on different transport systems (e.g., DDN and some other network) by a host on both systems.

SMTP is a specification only for the transfer of mail messages, not for their format, content, or interpretation. Thus, two computers with radically different mail systems, user interfaces, terminal characteristics, or other features, may have difficulty in exchanging messages that are intelligible to the final recipient without additional protocols.

SMTP is a good protocol for the ARPANET and similar DoD networks where a wide diversity of software development occurs and very little standard vendor software is used for applications like electronic mail. However, SMTP, like 1822 before it, is not likely to become a commercial standard. If DDN should evolve into a network system with some computers running vendor protocol architectures, SMTP may no longer be the electronic mail protocol of choice.

What are the alternatives? There are at least three standards activities of note:

(1) CCITT

(2) National Bureau of Standard

(3) IBM (de facto standards, but perhaps the most important)

The CCITT work is expected to result in the acceptance of a standard for "Message Handling Systems" in 1984, a specification of both a delivery standard and a user service. The CCITT standard calls for two protocols at level 7: a message transfer layer (analogous to SMTP), and a user agent level above that. The two layers can be compared to envelopes and letters, respectively. The first user agent (of several) is called

Interpersonal Messaging Services (IPMSs), and is designed for multimedia messages, including test, Teletex, and facsimile. The 1984 standard will not support full internal formatting of messages, but many coded fields are specified to provide for message handling (date, reply to, importance, etc.).

NBS has been a participant in the CCITT work, but has developed a separate standard which is now a FIPS adopted by several U.S. vendors. It remains to be seen if these two standards will be brought together.

Finally, IBM has developed its own standards in this area. It is possible and perhaps even likely that Document Interchange Architecture (analogous to SMTP and MT) and Document Content Architecture (analogous to UA/IPMSs) will be de facto standards of profound importance in communications. Already Wang and DEC have announced their support, and many other vendors are likely to follow.

What does this mean for the DDN? It is difficult at this time to predict which of these protocols will offer the lowest cost and greatest flexibility in the future. However, the IBM standard cannot be ignored because it should offer "complete" document intelligibility across a wide range of equipment, rather than to choose a more limited DoD standard such as SMTP, even though SMTP is designed for the DDN internet and DIA/DCA are not.

III. LOCAL AREA NETWORK SECURITY

A. INTRODUCTION

There is a rapidly growing proliferation of local area networks (LAN) throughout the Department of Defense (DoD). LANs represent the present state of the art in the continuing evolution of information and communications systems. LANs are playing a crucial role in the future of the DoD's major command and control and intelligence systems as well as local administrative, financial, personnel, logistics and word processing systems. There are plans underway to install literally miles of local area network cabling within the Pentagon in the very near future. Along with any new technology of this sort comes a vast array of compatibility, interoperability, and maintainability concerns and questions which frequently take years to resolve. One of the most serious of these concerns is security; yet it remains one of the most neglected aspects of system design.

Local area networks are frequently advertised with ads showing it is straightforward to link terminals or personal computers with all the other terminals and computers in local computer/communications system. But this presents a problem for DoD usage, since in almost every case users are only allowed access to some of the files and other terminals and computers. As a result, because the LAN connects all the computing resources in a local environment, concerns about the protection of sensitive information must be high on the DoD system designer's list, and yet frequently in the rush to get a working installation these concerns are overlooked or consciously ignored. The consequences of such action can be very serious. This chapter of this report will provide an overview of the security relevant aspects of local area network design and describe the vulnerabilities of these systems and some of the measures that can be taken today to protect them as well as areas where additional research work is required.

There have recently been several reports describing various security aspects of LANs (Refs. 9 & 10) which have made specific recommendations for those planning near-term implementations. This report will review those recommendations and update them with some very recent developments.

B. OVERVIEW OF LAN SECURITY VULNERABILITIES

The vulnerabilities of local area networks are best described within the overall context of the total systems in which they reside. Perhaps the best description of these vulnerabilities is contained in the Report of the Defense Science Board Task Force on Computer Security, published in February, 1970 (Ref. 11). This report begins with a description of the nature of the problem, including the types of computer systems and the threats to each of them. Such vulnerabilities as accidental disclosure, deliberate penetration, active infiltration, and passive subversion are discussed. In the areas of security protection the report describes typical physical protection measures, hardware and software leakage points, and communications and organizational leakage points. Figure III-1 is a reprint of the computer network vulnerability chart which appeared in this report. This figure describes the various types of security failures that can occur in a computer communications system.

When considering the vulnerabilities of local area networks in the context of Fig. III-1, one can view the LAN as replacing

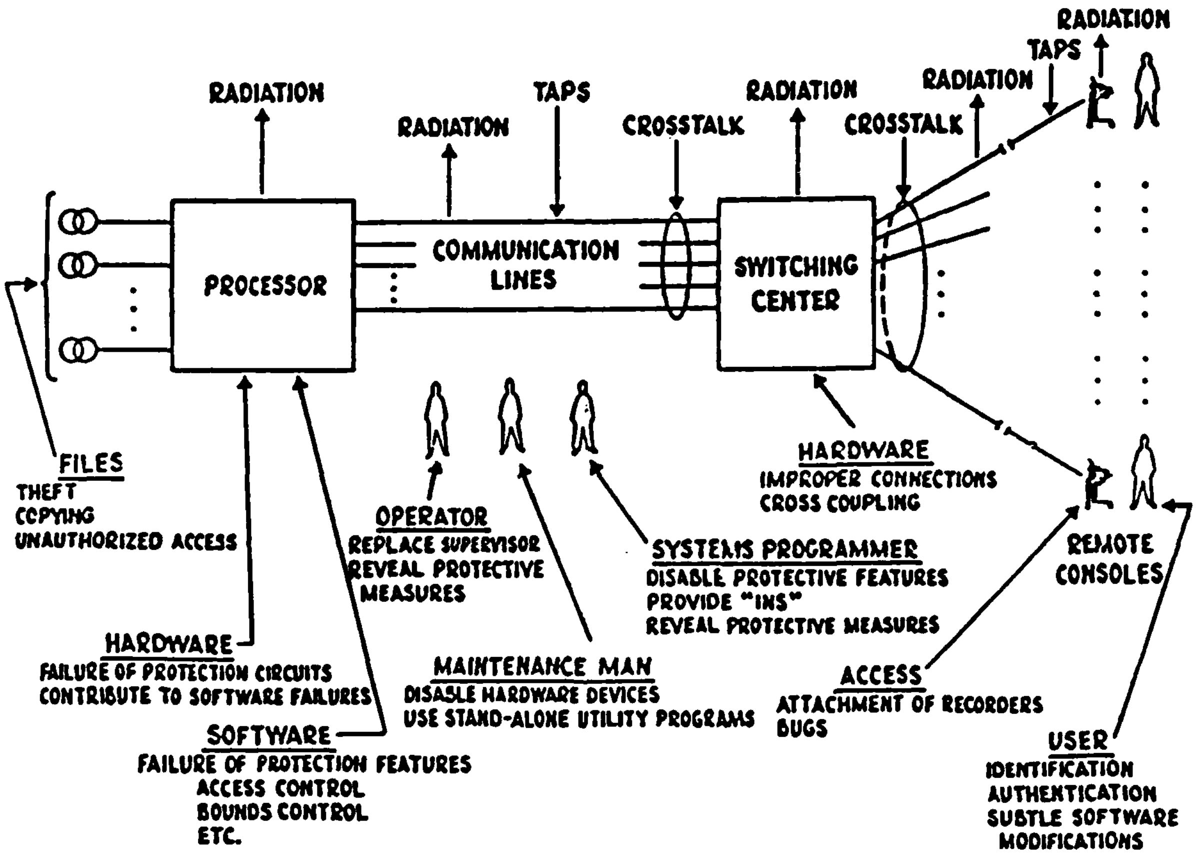


FIGURE III-1. Computer Network Vulnerabilities
Source: (Ref. 11)

the set of wires from the switching center to the individual users, and/or replacing the communication lines between the processor and the switching center or replacing the entire switching center itself. The LAN is therefore subject to wire tapping, radiation, crosstalk, hardware and software failures, as well as personnel vulnerabilities from maintenance people, operators, and system programmers.

There are many technologies for achieving local area network capabilities. These include the use of broadband, cable TV systems, baseband coaxial systems, and private branch exchange (PBX) switching centers. The transmission media extend from twisted pair, to coaxial cables, to fiber optic cables. Many require a new physical plant but some allow extensive use of existing facilities.

There are significant examples of the evolution of local area network usage in major programs of the Department of Defense. For example, the WWMCCS Information System (WIS) Modernization Program has as its key initial ingredient the installation of a local area net facility at each WWMCCS site, linking user workstations to the Honeywell 6000 computers and the WWMCCS Intercomputer Network. As the modernization program proceeds, additional resources will be attached to these LANs, providing special functions which are now performed by the Honeywell 6000 computers. Eventually, when all of the processing now being done on the H6000s has been migrated away, they will be removed from the network.

Most of the WWMCCS sites are at the Top Secret classification level and all users presently having access to the H6000 machines must be cleared to that level, even though close to 85 percent of the information on those computers is classified only at the Secret level. Therefore, a major factor in the evolution of the WIS program is the establishment of a capability to operate at several different classification levels and eventually to allow the use of multilevel secure computer systems.

In this case, since the LAN provides the vital link between all user workstations and computational resources, it is absolutely essential that the LAN be able to properly protect classified information passing through it. This could be done by providing different local area net facilities for each classification or sensitivity level within a WWMCCS site. However, such a solution would be very expensive, administratively burdensome, and unacceptable operationally. It would be very advantageous if the LAN supplied with the WIS modernization were able simultaneously to handle sensitive information at several different levels. Some recent developments in LAN technology allow for at least limited service of this type with evolutionary expansion possible. These techniques will be described later in this chapter.

At the other end of the sensitive information handling environment are the efforts to install local area networks to handle the proliferation of unclassified terminals accessing local computer systems. There are major efforts at most DoD facilities to upgrade existing communication services to local area net capabilities. The thrust of these efforts is to allow the linkage of existing and future unclassified computer services to the wide variety of terminal systems scattered throughout a facility.

Frequently there is little or no concern expressed for the protection of sensitive information on these networks. The excuse is given that the network contains only unclassified information and if at some point in the future it should become necessary to handle sensitive information, some form of encryption device can then be added to the network. This approach is unfortunate and dangerous from several perspectives. First, even though individual pieces of information on these data base systems may be considered unclassified, frequently the aggregation of personnel, financial, or logistics information becomes quite sensitive, if not actually classified. Many times, for privacy reasons alone, personnel information should receive some degree of sensitive protection. Financial information can, in the wrong hands, be manipulated to cause serious losses.

The other aspect of this excuse, which is most unfortunate, is that with proper beforehand design efforts, sensitive information can be protected on an otherwise simple LAN structure. Conversely, it is impractical to contemplate the future addition of encryption or some other form of protection to a LAN which was not properly designed to handle such a capability in the first place.

There is an interesting problem that evolves when designers upgrade unclassified, nonsensitive information services without giving similar capabilities to important and sensitive functions. The effect of this is that the routine administrative procedures in an organization become highly automated and efficient, while the important planning, financial, logistics, and personnel functions remain in essentially isolated and frequently manual environments. When this situation arises there is a very dangerous tendency to go ahead and use the unprotected LAN facility to "get the job done."

For all of the above reasons it is important that the security/sensitive information handling requirements of a particular installation be considered up front prior to the establishment of the specifications for a particular local area network.

C. SECURITY-RELEVANT CHARACTERISTICS OF LAN

A local area network connects communicating devices that are not far from each other but where there is a definite physical separation ranging from a few dozen meters to a few kilometers. In most cases a single coaxial cable replaces an entire network of conventional wires and permits simultaneous two-way communications among communicating devices. Frequently this system will support hundreds of high-speed digital channels with text, data, and in some cases voice, and image signals. Many of the characteristics of LANs that make them highly attractive from an installation ease and flexibility point of view have very negative connotations from a security perspective. Some of these characteristics, excerpted from an excellent reference text on LANs, "Designing and Implementing Local Area Networks," (Ref. 12), will be described now.

In a typical coaxial broadband local area network, radio frequency (RF) signals are transmitted over trunk and feeder lines to drop cables which provide links to the user outlets. The trunk is usually a protected cable employing rigid aluminum shielding with a bending radius ten times the diameter of the cable. The feeder cables and drop cables are much smaller and more flexible.

Taps can be placed anywhere along the feeder cable to provide drop cable connections to outlets. The drop cables and feeder cables typically incorporate foil and braid shielding to minimize radiation leakage among cables in close proximity to each other. Directional couplers divide or combine inputs and outputs of RF signals. They are used to ensure that signals being transmitted from any network device will be routed only to the head end device. One of the key advantages of using coaxial cable in systems like this is its ability to support multiple taps. With the isolation provided, each outlet stands alone, and the connection or disconnection of a user device has no effect on the operation of the overall system. Broadband coaxial systems employ components that are readily available from community antenna TV (CATV) or cable TV systems. These systems are characterized by their ease of connecting new terminal ports simply by adding a tap. It is also easy to move devices anywhere on the network since all devices receive all of the information transmitted over the network.

As can be seen by the above discussion, most of these features provide advantages for the easy installation and operation of a system. But many of them, in particular the ease of attaching taps in an undetected manner and the fact that all information passes each point on the network, constitute serious concerns from a security point of view.

Fiber optics is frequently proposed in lieu of coaxial cable in LANs. The characteristics of fiber optics, which are its principle advantages, include higher bandwidth, immunity to electrical interference, lack of electromagnetic radiation, and smaller physical diameter characteristics. However, fiber optic systems are more difficult to tap, and amplifiers and related equipment are frequently much more expensive.

Another form of local area network that is evolving in some arenas is the computer-based private branch exchange (PBX). Such systems provide data connectivity for terminal-to-host traffic in addition to the normal voice service. The advantage of such a service is that it makes use of the existing wire plant, which minimizes the changes to existing facilities. The disadvantages include relatively low bandwidth and difficulty providing host-to-host high-performance links. From a security viewpoint, this type of service is little different than that associated with normal telephone service. Because this form of LAN implementation is not particularly useful in systems like WIS, it will not be considered further here.

D. PROTECTION MEASURES

This section will describe a series of protection measures that are available for local area network security. It includes an analysis of the overall security of the information system of which the LAN is a part. In this context the physical, administrative and personnel security measures for the LAN are closely linked to those procedures associated with individual terminals and host computers which comprise the total system. The local area network must receive the same protection measures afforded to the highest level of sensitive information contained in the overall system.

In the case of WWMCCS, for example, the highest level of classified information is Top Secret and all terminal, host, and communications functions, including the LAN, must be protected to that level. This implies that all personnel must have Top Secret clearances, that all terminals and hosts must be located in Top Secret cleared facilities, and that all communications must be contained either within Top Secret facilities or be protected via military-approved cryptographic devices for handling Top Secret information. Once the WIS system evolves to controlled mode with some users only cleared to Secret, the terminal facilities for those users need only be cleared for Secret information. But, to the extent that the communications linking them may also contain Top Secret information (i.e., on a local area network), those communication links must continue to be protected to the Top Secret level.

There are, in general, two ways to protect a local area network system for handling classified information. The first is to physically contain the transmission media (e.g., coaxial or fiber optic cable) in a protected wire line environment. The second approach is to provide some form of encryption for the communications passing over this LAN media. Each of these will be considered in detail now.

Protected wire line systems typically are quite expensive to install and they have limited flexibility for attaching additional feeder circuit and drop lines. These systems range from simple welded conduit for use with information with a low level of sensitivity to sealed, pressurized enclosures which have automatic pressure loss detection devices to alarm in case the conduit is broken, accidentally or maliciously, to systems where the conduit must remain under visual scrutiny of security officers at all times.

In many cases buildings which were designed for handling sensitive information via communications media will have conduits built into the floor or ceiling which will serve as protected wire line environments. In these cases it is merely necessary to extend the drop line from conduit to the actual communicating device via shielded coaxial cable or similar means. The functions of the protected wire line are to preclude tampering, wire tapping, or line monitoring via direct connection or through the detection of compromising emanations (TEMPEST).

For unclassified information it may be sufficient to enclose the LAN transmission media in an aluminum conduit with sealed connections. This type of protection is frequently provided just for the physical protection of the cable from being damaged or broken through normal building use. The requirements for additional protected wire line measure for protecting classified information are detailed in individual service security regulations and specifications. For new installations, the inclusion of a protected wire line facility is not nearly as significant an expense as in older facilities.

The second alternative for protecting classified information on a LAN is to provide some form of encryption to the information. Once encrypted, the information is considered unclassified by virtue of having been subjected to the encryption algorithm. Anyone intercepting the encrypted information would have great difficulty in obtaining the sensitive information contained therein.

The simplest form of encryption which is normally provided to unprotected communication channels is link encryption, in which the data is encrypted as it enters the transmission medium (i.e., just after the modem device). Unfortunately, due to the high bandwidth of all LAN media, link encryption is not practical. In the case of baseband systems, data rates of 2 to 10 Mbps are beyond the capability of present encryption devices, except for very expensive equipment. Broadband systems which use even higher analog signals are impractical to encrypt. Given that a pair of link encryption devices would be needed at each outlet of the LAN, the cost of such a service quickly becomes excessive, even if equipment were available today.

A second form of encryption protection for a LAN is the use of end-to-end encryption. In this case the information to be transmitted is encrypted prior to being entrusted to the network and remains encrypted throughout its transmission, being decrypted only upon exiting the LAN interface device. This is a more practical form of protection both from a data rate and cost point of view. The data rates are those of the individual communicating devices and not the aggregate rate of the transmission media itself. In this case the number of devices needed corresponds to the number of drops or communicating devices on the LAN.

The basic problem with this approach is: there are no devices available with this capability. The Blacker program is attempting to provide this type of service on a wide area network such as the Defense Data Network (DDN). Assuming it can be made to work there, it is reasonable to assume that one could extend this capability out to a LAN connected to the DDN. However, initial Blacker devices for the DDN are not expected until 1988 and probably will not be available in production quantities until the early 1990s. The present contract does not call for development of LAN devices although such devices are anticipated to be needed shortly after the initial DDN devices.

End-to-end encryption provides protection for the data being transmitted, but address information, indicating where that data is to be sent on the network, must be kept in the clear. This dual function for the encryption device makes it much more complex than a simple link encryption facility. In addition, some form of key distribution center and access control mechanism is required for anything other than the simplest form of end-to-end encryption. Unfortunately, the procedures for doing key distribution are not fully understood. They are being developed as part of the Blacker program but will not be available in LAN environments until the early 1990s.

While encryption and, in particular, end-to-end encryption shows real promise for providing excellent protection for sensitive information on a LAN, the difficulties to be overcome prior to availability coupled with the cost of the devices themselves, including the key distribution and access control mechanisms, indicated that it may be eight to ten years before this type of protection will be generally available. Therefore, anyone considering LAN service before that will have to provide protection via some form of protected wire line facility.

E. TRUSTED LOCAL AREA NETWORK CAPABILITIES

Because of the problems inherent in applying encryption techniques to LANs and the apparent delay in having such resources available for the next five to eight years, system designers have turned to other means of providing security. One approach which is straightforward but very limited in its operational characteristics is to install a separate physical LAN for each level of sensitive information. While this approach is immediately available using standard components, it has the severe disadvantages of requiring considerable duplication of equipment, extensive additional physical construction of protected wire line facilities, and complex operating procedures to ensure that the user employs the proper terminal and LAN for the appropriate level of security. In systems that involve a reasonable range of sensitive information, this operational complexity may pose such a large security risk that even this low technology approach may be unacceptable.

Several years ago consideration began to be given to finding means to assure that at least the addressing portions of the LAN could be made highly reliable. If the LAN must be enclosed in a physically secured, protected wire line conduit anyway, and if some means could be found for ensuring that the LAN would deliver the information to the proper destination without modification, then simultaneous operation of a LAN with multiple levels of sensitive information could be achieved. The term Trusted LAN was applied to this concept and several studies were initiated to evaluate its feasibility (Refs. 9 & 13).

The simplest form of this trusted LAN is shown in Fig. III-2. In this case the LAN is considered multilevel secure, that is, capable of interfacing terminals and host computers that operate at multiple levels of classification. The LAN interface unit is "trusted" to properly label information entering the network as to its sensitivity and to ensure that information is delivered from the network to destinations at the same sensitivity level.

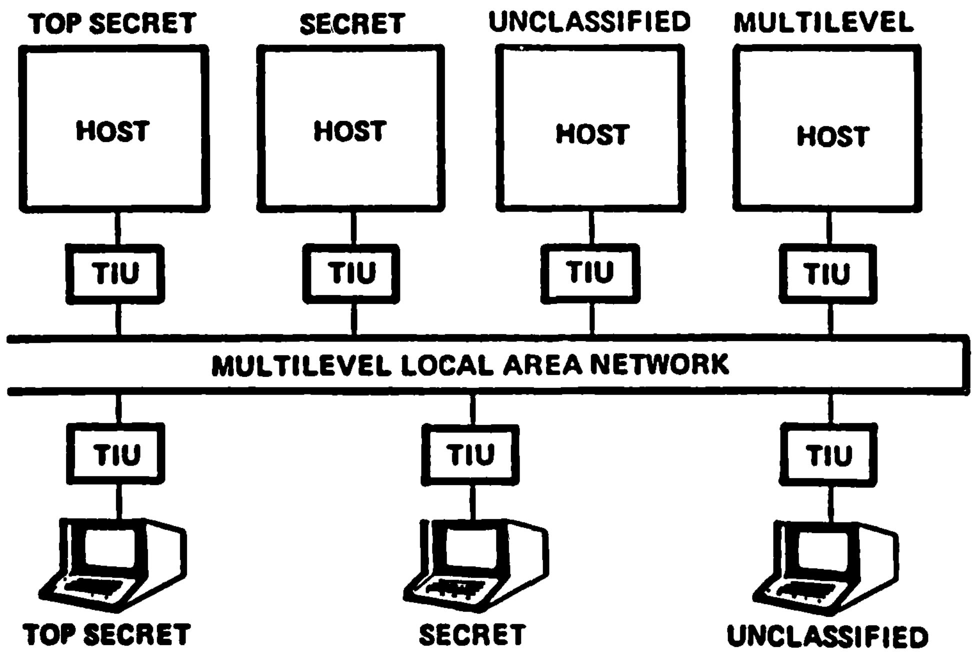


FIGURE III-2. Simple multilevel local area network
Source: Ref. 13

Early analyses of this approach concluded that a trusted interface unit would have all of the development and verification complexity of earlier attempts to build trusted general purpose operating systems. However, more detailed examination shows that much of the complexity that is inherent in trusting such operating systems comes from the need to enforce rather complex security policies that allow users at certain levels in the security hierarchy to read and write data at their present level, to read but not write data at levels below their present level, and to (theoretically at least) write but not read data above their present level. This security policy was first stated mathematically by Bell and LaPadula in 1974 and remains the basis for policy enforcement in operating systems today.

However, examining the function which is being performed by the LAN in these scenarios shows that the LAN does not need to enforce a policy as complex as Bell and LaPadula. Rather, the LAN should be viewed as a logical replacement for a bundle of wires directly linking every terminal to every host computer in a local area. Under this view, the trusted LAN must be shown to enforce the much simpler policy stated above of properly labeling information on entry to the network and ensuring delivery only to destinations approved for handling similar sensitive information.

It is interesting to note that the fear of the complexity of trusted LANs that drove many to favor the use of end-to-end encryption techniques can be misleading, since in order to handle properly the bypassing of address information in the end-to-end device, a degree of trust equal to or exceeding that of the trusted LAN is required. It is just this issue that is at the core of the difficulty being faced by the Blacker program.

Figure III-3 is a more complex situation involving several physical environments at different security levels with both trusted and untrusted interface units, encryption between the physically protected LANs and host computers operating at all levels of classified information.

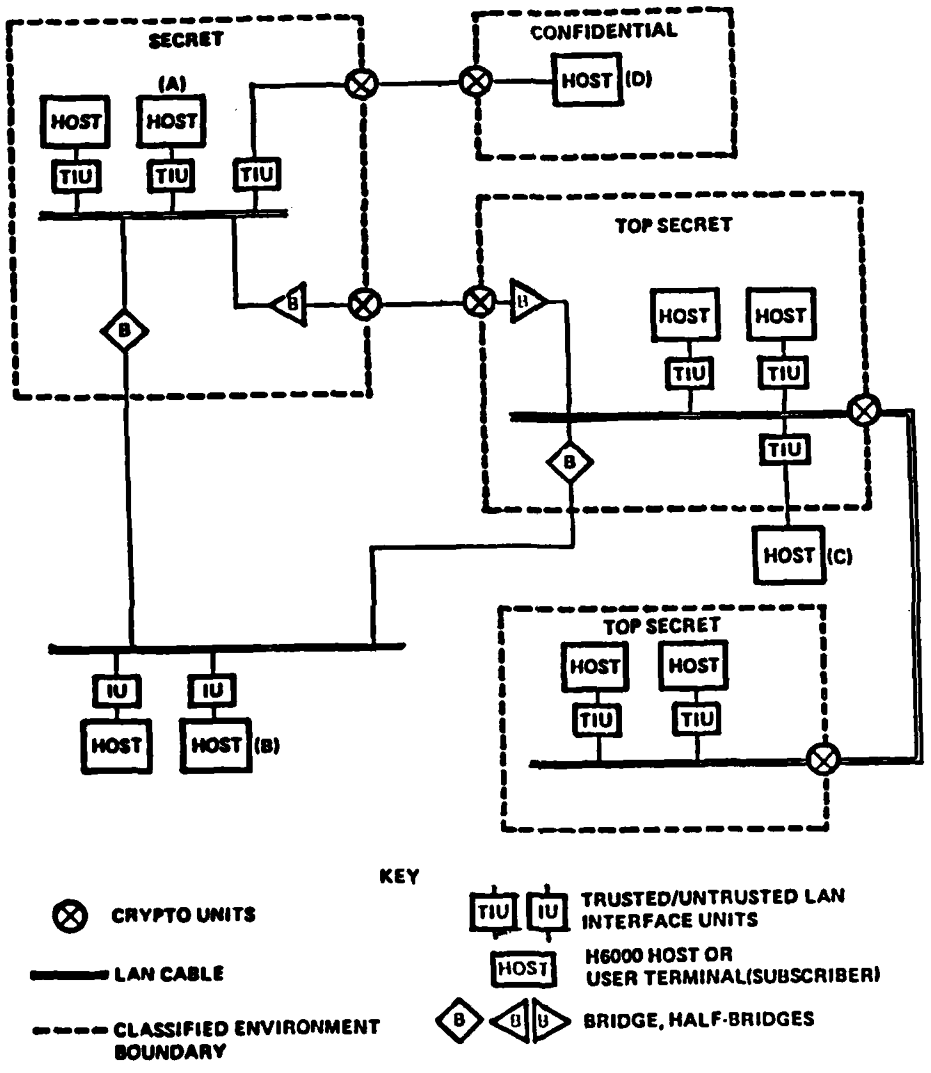


FIGURE III-3. Full Multilevel Local Area Network
Source: Ref. 13

In a report entitled "Cable Bus Applications in Command Centers - Security Issues" (Ref. 10), Robert Shirey describes the WWMCCS LAN requirements and details the use of encryption in considerable detail including an analysis of the potential use of public key cryptography. This report concludes that "multilevel secure LANs can and should be built using Blacker end-toend encryption," while cautioning that unless action is taken soon (the report date is February 1982), "it is almost certain that the cryptors will not be available when needed." The report acknowledges that local area networks will need both encryption and trusted software.

A second report on this same area, "WIS Local Area Network Issues," by William H. Blankertz and David A. Gomberg (Ref. 9), describes the LAN security issue in the following terms:

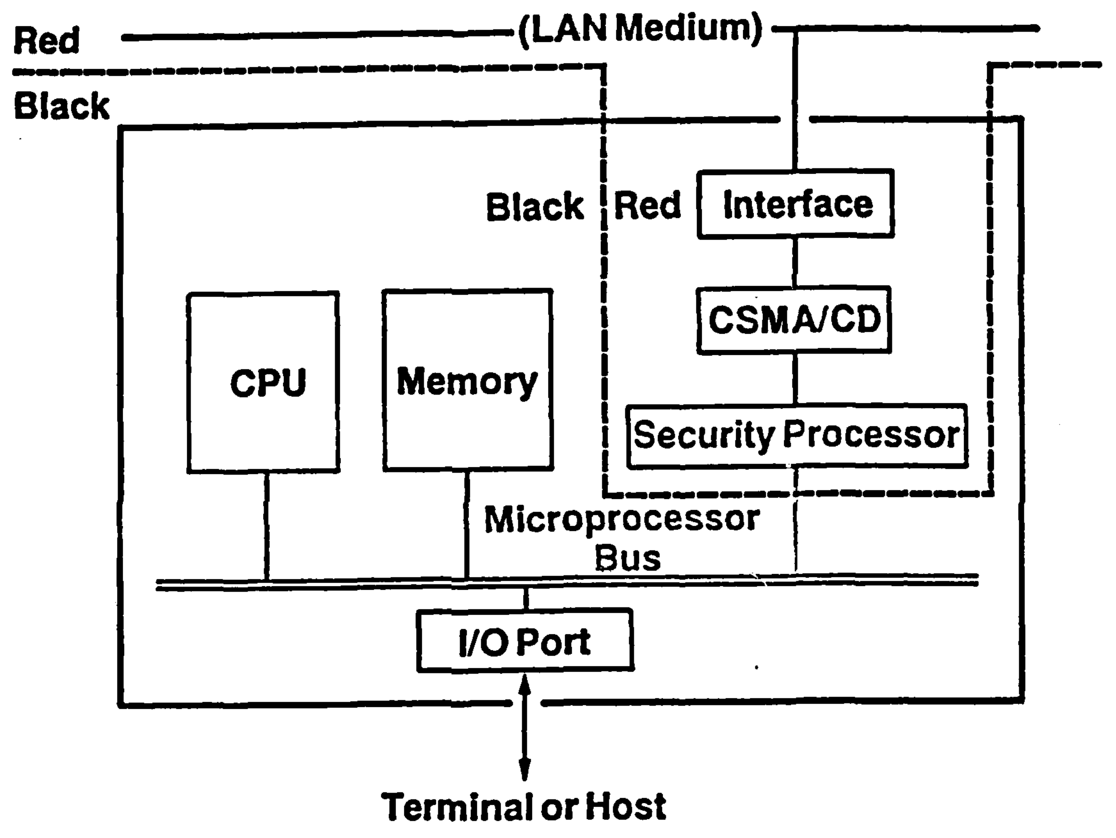
> "By far the greatest risk in the development of WIS LAN is the requirement that the LAN provide multi-level security. It is the expressed opinion of many WWMCCS users that the WIS LAN will be of little utility without MLS. Yet there is no assurance that MLS is achievable; there are no clear-cut solutions to the technical problems that MLS poses. A successful approach to MLS will most likely require features from ... encryption, trusted software and physical protection."
>
> "The LAN security architecture intersects with almost all aspects of the WIS design. ... Problems such as cryptor availability and lack of adequate tools to develop and verify trusted software may cause considerable delay in the MLS LAN implementation."

A description of the issues surrounding trusted LANs is given in "Design for a Multilevel Secure Local Area Network" by Deepinder P. Sidhu and Morrie Gasser (Ref. 13). Written again from the context of the WWMCCS environment, this report describes the multilevel secure LAN problem, and details the characteristics of several alternative solutions. The report then proposes several operational scenarios for the use of trusted interface units, bridges, gateways, and guard systems. The report proposes a series of incremental upgrades starting from a single subnetwork operating in a system-high, physically protected environment, expanding to trusted interface units able to support variable-level terminals and controlled and multilevel mode hosts.

Figure III-4, from the above report, shows a logical structure for a trusted interface unit. The LAN medium, interface, Carrier Sense Multiple Access with Collision Detection (CSMA/CD) and security processor components are all in the physically protected areas denoted as RED. These components all must be trusted to operate correctly. Assuming that this can be done and that these components can be isolated from modification by other portions of the interface unit, then the other components do not have security relevance and therefore need not be trusted.

F. RECENT DEVELOPMENTS

There are a number of efforts underway to attempt to resolve the local area network security issues. The one that appears furthest along and of most potential value to WWMCCS and other similar programs is the System Development Corporation (SDC) MIL/INT product line which was first described on September 30, 1983 and is expected to be announced formally, with prices and quantity delivery, before June 1984. This local area network system (Fig. III-5) is a broadband cable with a 2-Mbps signalling rate. The SDC product line consists of eight devices ranging from host front ends and terminal concentrators (for both the LAN and the Defense Data Network (DDN)) as well as intersite, interchannel, and DDN gateways and a network control center.



Source: Sidhu and Gasser, "Design for a Multilevel Secure Local Area Network" (Ref. 13)

FIGURE III-4.

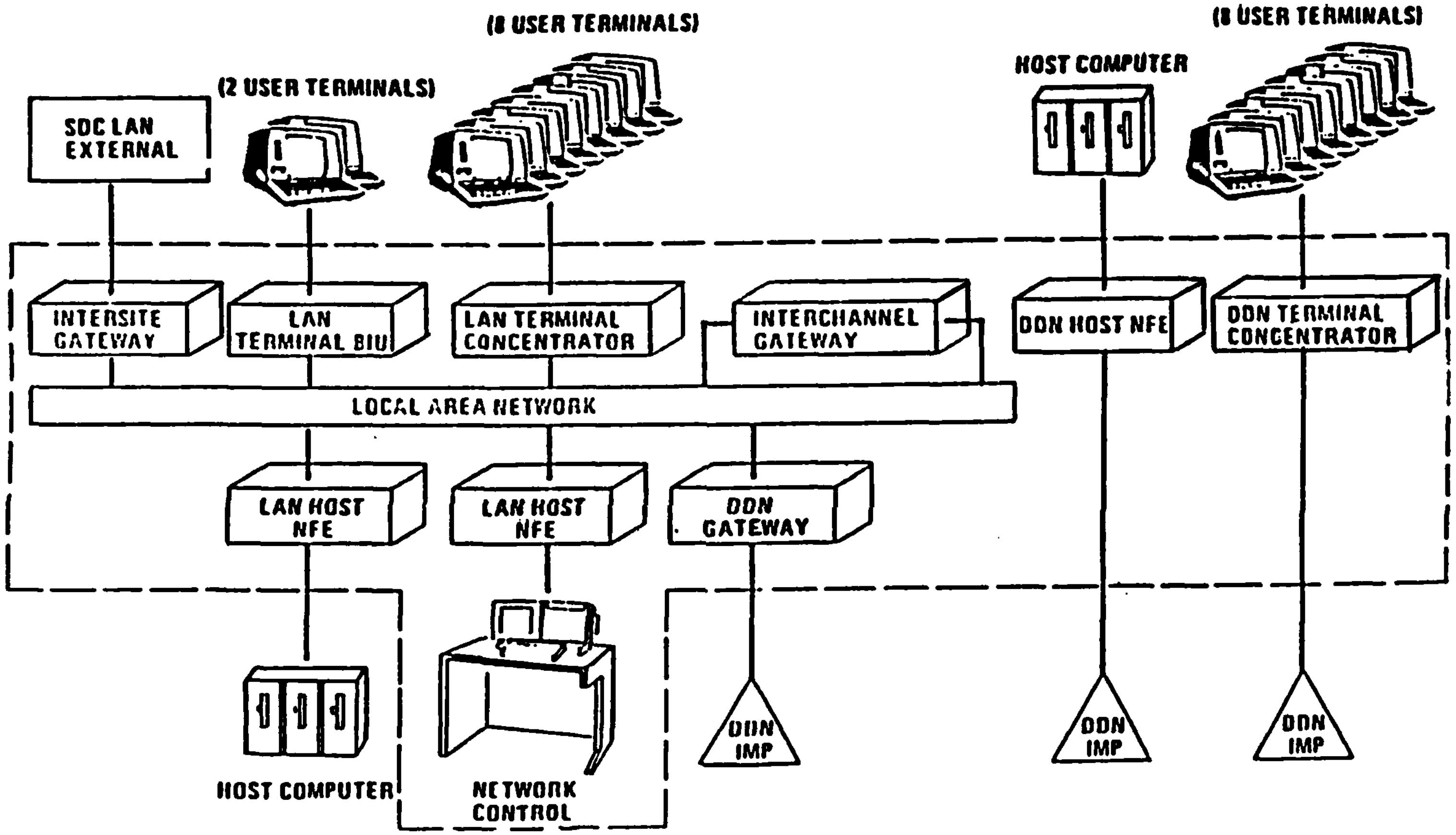


FIGURE III-5. SDC MIL/INT Network Product Line Overview

The devices are designed to work specifically with the DDN and therefore implement the full set of DoD Standard Protocols including the Transmission Control Protocol/Internet Protocol (TCP/IP), the Network Virtual Terminal (NVT) protocol, and the Host Front End (HFE) Protocol. One of the major advantages of this LAN is the ability of any host or terminal to easily access other subscribers on this net or on other nets across the DDN through the use of these protocols.

The other interesting characteristic of this soon-to-be-announced product is a series of security-related enhancements that will implement the features of a trusted LAN. In the first step, a label-checking mechanism will be added to the BIU at exactly the place where the security processor is located in Fig. III-4. Information entering the network from a user terminal or host will have its security label (the security field in the internet header) checked to ensure that it is appropriate for the security level of that site. Information leaving the network will also be checked for appropriate security level prior to being allowed into the BIU for processing.

The initial label-checking mechanism is being implemented in firmware immediately adjacent to the CSMA/CD unit to minimize the need to verify its performance and to trust other portions of the BIU. As a result there is limited flexibility in identifying large numbers of possible communities to which a terminal or host may belong. Future expansion of this mechanism, including its implementation on an Intel 80286 processor board, as part of a planned performance upgrade, will allow much more granular security-level identification.

SDC is actively promoting the MIL/INT LAN for use in WIS and other DoD environments. A version of MIL/INT is already being installed in a DODIIS environment. The combination of support for the complete DoD protocol suite and the inclusion of mechanisms for label checking make this product an important addition to the available local area network capabilities. It is possible that this product line and those of other manufacturers now in development may make possible multilevel secure local area networks without requiring the long delays anticipated in achieving end-to-end encryption for LANs. Our study of these possibilities is continuing.

IV. LOGISTICS NETWORKS

A. BACKGROUND

The study of the logistics data communications networks began with a study of the logistics usage of the AUTODIN I system. This system was originally designed for logistics usage but has subsequently grown to support other users. Nevertheless, about 50-60% of its total usage was estimated to be by logistics users. We therefore contacted personnel in the Army, Navy, and Air Force Logistics Commands to get estimates of their usage and plans.

From our work with them, it is apparent that the AUTODIN system is in danger of becoming overloaded with logistics traffic. This conclusion is supported by such facts as the overloading of the AFLC operation (by Thursday of each week they often have a backup of 40,000-60,000 line blocks which requires the entire weekend to run off), the running backlog at the Presidio, and problems with Battlecreek DLA overloading the system. JCS concerns about the impact of logistics traffic during crisis situations and the resultant effect on Command and Control activities increases the need for immediate action.

The Services also reported a need for future service in excess of what can be projected as reasonable for AUTODIN. (The Air Force predicts a growth of 15 percent a year, for example, and the Army 20 percent a year.) Since logistics traffic is currently estimated at 50-60 percent of AUTODIN traffic, this will result in a disproportionate share of AUTODIN traffic going to logistics users.

In order to define further the traffic flow, meetings were then held with DIA personnel. In order to give some feeling for DIA usage, Figure IV-1 shows some DIA transmit and receive statistics.

In addition, there is a large volume of data transferred by mail and courier. This data was, at one time, principally punched cards but is now transported primarily on magnetic tape. Many users would prefer using telecommunications but the lack of availability of lines, automated systems, and cost has led to the slower mail and courier method.

Much of this data is transferred for contingency operations and this data has been broken out as it becomes most important from a mobilization viewpoint. Figure IV-2 shows some figures for the types of data transfer, giving an idea of the volumes involved. As an example, the New Cumberland Army Depot generates on magnetic tape enough data to fill a 50-kb line full time for 22 days (of 24 hours) in each quarter. The contingency data alone would require about 5 days (of 24 hours) for transfer at 50 kb each quarter.

If the system involved were much more interactive, then the total data transferred could be reduced, since many file transfers are now being made and queries could replace some of these; however, it is clear that a large volume of data is lurking in the background in this form in addition to the data now being transferred using telecommunications.

Further investigation of the source of the AUTODIN problem specifically was then made. The majority of the traffic was found to center around two DLA sites, the DAAS at Gentile and the DAAS at Dayton, Ohio. Some representative figures for those sites can be found in Figure IV-3. When a requisition for materiel is made, the AUTODIN system is used and the order is sent to one of the two DAAS sites. The DAAS site then determines which inventory control point has jurisdiction and sends the order to that inventory control point. Inventory

SITE (CENTERS & DEPOTS)**	MONTHLY AVERAGE MESSAGES	MONTHLY AVERAGE LINE BLOCKS	MONTHLY PEAKS MESSAGES	MONTHLY PEAKS LINE BLOCKS	*AVERAGE PER HOUR MESSAGES	*AVERAGE PER HOUR LINE BLOCKS	AVERAGE PEAK HOUR LINE BLOCKS	AVERAGE PEAK DAY LINE BLOCKS
DGSC Richmond, Va	13,482	1,387,842	15,429	1,882,068	18.6	1,917	3,469	138,754
DDMT Memphis, TN	3,871	946,662	4,406	1,114,783	5.3	1,307	3,054	73,298
DASC Cameron Station, VA	2,940	91,865	4,151	131,268	4.1	127	1,144	27,466
DCSC Columbus, OH	19,347	2,800,579	24,410	3,212,803	26.9	3,868	10,700	256,804
DPSC Philadelphia, PA	16,482	2,511,420	24,536	3,161,152	22.8	3,469	12,678	304,223
DDMP Mechanicsburg, PA	1,806	559,776	2,058	629,055	2.5	773	2,658	63,791
DISC Philadelphia, PA	15,414	1,299,914	24,098	1,592,005	21.3	1,795	8,600	206,394
DISC Philadelphia, PA	9,717	1,197,579	15,915	1,476,202	13.4	1,654	8,315	199,565
DLSC Battle Creek, MI	23,952	4,419,059	40,948	6,229,459	33.1	6,104	15,974	383,366
DLSC Battle Creek, MI	25,584	4,899,512	42,901	7,337,603	35.3	6,767	19,875	477,002
DLSC Battle Creek, MI	22,482	3,925,038	44,833	5,522,380	31.0	5,421	16,019	384,450
DLSC Battle Creek, MI	21,700	3,810,416	38,588	5,690,508	30.0	5,263	14,342	344,196
DDOU Ogden, UT	2,858	880,914	3,219	977,610	3.95	1,217	2,437	58,489

*Estimated
**Centers and Depots work 24 hours per day, 7 days a week

FIGURE IV-1a. Transmit Statistics--January 1982-June 1982

SITE (DCASRS)**	MONTHLY AVERAGE MESSAGES	MONTHLY AVERAGE LINE BLOCKS	MONTHLY PEAKS MESSAGES	MONTHLY PEAKS LINE BLOCKS	*AVERAGE PER HOUR MESSAGES	*AVERAGE PER HOUR LINE BLOCKS	AVERAGE PEAK HOUR LINE BLOCKS	AVERAGE PEAK DAY LINE BLOCKS
DCRI Chicago, IL	2,391	85,124	2,892	97,062	14.2	507	2,107	16,853
DCRS St. Louis, MO	1,887	60,195	2,483	85,256	11.2	358	1,247	9,975
DCRO Cleveland, OH	2,057	78,518	2,606	95,327	12.2	467	1,707	13,654
DCRA Atlanta, GA	2,031	69,817	2,496	97,568	12.1	416	1,767	14,133
DCRN New York, NY	2,531	90,193	2,749	100,475	15.1	537	2,707	21,657
DCRP Philadelphia, PA	4,404	630,988	5,832	932,948	26.2	3,756	10,271	82,169
DCRB Boston, MA	3,084	122,610	3,506	138,382	18.4	730	3,132	25,052
DCRL Los Angeles, CA	5,912	251,081	9,943	448,298	35.2	1,494	5,617	44,938
DCRT Dallas, TX	2,429	126,936	2,791	16,228	14.5	756	1,879	15,030

*Estimated
**DCASRs work 8 hours per day, 5 days a week

FIGURE IV-1b. Transmit Statistics--January 1982-June 1982

SITE (CENTERS & DEPOTS)**	MONTHLY AVERAGE		MONTHLY PEAKS		*AVERAGE PER HOUR		AVERAGE PEAK HOUR	AVERAGE PEAK DAY
	MESSAGES	LINE BLOCKS	MESSAGES	LINE BLOCKS	MESSAGES	LINE BLOCKS	LINE BLOCKS	LINE BLOCKS
DGSC Richmond, Va	23,083	1,399,552	26,673	1,699,302	31.9	1,933	4,777	114,639
DDMT Memphis, TN	6,745	816,236	7,817	944,779	9.3	1,127	2,744	65,852
DASC Cameron Station, VA	7,883	439,665	11,457	533,661	10.9	607	1,873	44,952
DCSC Columbus, OH	27,433	1,809,195	33,344	2,082,710	37.9	2,499	13,232	317,561
DPSC Philadelphia, PA	23,483	1,802,947	30,697	2,350,798	32.4	2,490	7,082	169,965
DDMP Mechanicsburg, PA	3,412	497,029	4,109	616,159	4.7	686	2,652	63,640
DISC Philadelphia, PA	23,941	2,438,446	33,690	3,423,039	33.1	3,368	8,818	211,628
DISC Philadelphia, PA	22,315	498,071	27,092	593,829	30.8	688	2,868	68,824
DLSC Battle Creek, MI	9,500	1,266,964	11,273	1,567,306	13.1	1,750	7,689	184,547
DLSC Battle Creek, MI	1,958	805,509	2,466	1,078,602	2.7	1,112	6,116	146,776
DLSC Battle Creek, MI	4,874	810,651	6,835	1,027,983	6.7	1,120	3,822	91,716
DLSC Battle Creek, MI	1,659	422,617	2,243	660,827	2.3	583	2,673	64,152
DDOU Ogden, UT	4,756	667,539	5,642	791,920	6.57	922	2,050	49,214

*Estimated
**Centers and Depots work 24 hours per day, 7 days a week

FIGURE IV-1c. Receive Statistics--January 1982-June 1982

SITE (DCASRS)**	MONTHLY AVERAGE		MONTHLY PEAKS		*AVERAGE PER HOUR		AVERAGE PEAK HOUR	AVERAGE PEAK DAY
	MESSAGES	LINE BLOCKS	MESSAGES	LINE BLOCKS	MESSAGES	LINE BLOCKS	LINE BLOCKS	LINE BLOCKS
DCRI Chicago, IL	3,691	196,236	4,248	228,427	22.0	1,168	2,442	19,538
DCRS St. Louis, MO	1,492	89,272	1,963	132,699	8.9	531	2,604	20,830
DCRO Cleveland, OH	1,637	105,068	2,037	123,279	9.74	625	2,388	19,104
DCRA Atlanta, GA	1,790	139,227	2,391	224,179	10.6	829	2,564	20,510
DCRN New York, NY	3,030	124,289	3,601	154,296	18.0	740	2,435	19,481
DCRP Philadelphia, PA	5,437	284,866	7,243	373,088	32.4	1,696	4,790	38,324
DCRB Boston, MA	2,955	132,714	3,365	164,420	17.6	790	1,729	13,830
DCRL Los Angeles, CA	4,386	234,250	7,515	442,691	26.1	1,394	3,964	31,708
DCRT Dallas, TX	1,790	126,601	2,133	151,210	10.6	754	2,295	18,358

*Estimated
**DCASRs work 8 hours per day, 5 days a week

FIGURE IV-1d. Receive Statistics--January 1982-June 1982

SUMMARY: MAILED/COURIERED DATA PRODUCTS 1982

Magnetic tape data transmitted by mail or courier (Equated to 80 Character Records) per quarter:

Magnetic Tape (COOP)	30,240,000
Magnetic Tape (Regular)	12,513,187
Total Per Quarter	42,753,187
Microfiche Per Quarter	None Reported
Hard Copy: Pages Per Quarter	None Reported

FIGURE IV-2a. Installation: Rock Island Arsenal (ARRCOM)

Magnetic tape data transmitted by mail or courier (Equated to 80 Character Records) per quarter:

Magnetic Tape (COOP)	128,600,111
Magnetic Tape (Regular)	14,263,277
Total Per Quarter	140,863,388
Microfiche Per Quarter	1,525
Hard Copy: Pages Per Quarter	895,809

FIGURE IV-2b. Installation: Letterkenny AD (DESCOM)

Magnetic tape data transmitted by mail or courier (Equated to 80 Character Records) per quarter:

Magnetic Tape (COOP)	36,672,000
Magnetic Tape (Regular)	15,704,000
Total Per Quarter	52,376,000
Microfiche Per Quarter	None Reported
Hard Copy: Pages Per Quarter	None Reported

FIGURE IV-2c. Installation: Fort Monmouth (CECOM)

SUMMARY: MAILED/COURIERED DATA PRODUCTS 1982

Magnetic tape data transmitted by mail or courier (Equated to 80 Character Records) per quarter:

Magnetic Tape (COOP)	None Reported
Magnetic Tape (Regular)	330,248
Total Per Quarter	330,248
Microfiche Per Quarter	492
Hard Copy: Pages Per Quarter	9,963

FIGURE IV-2d. Installation: Aberdeen PG (TECOM)

Magnetic tape data transmitted by mail or courier (Equated to 80 Character Records) per quarter:

Magnetic Tape (COOP)	19,040,000
Magnetic Tape (Regular)	4,724,788
Total Per Quarter	23,764,788
Microfiche Per Quarter	862
Hard Copy: Pages Per Quarter	20,557

FIGURE IV-2e. Installation: TSARCOM

Magnetic tape data transmitted by mail or courier (Equated to 80 Character Records) per quarter:

Magnetic Tape (COOP)	99,298,014
Magnetic Tape (Regular)	49,346,466
Total Per Quarter	148,644,480
Microfiche Per Quarter	None Reported
Hard Copy: Pages Per Quarter	-------------

FIGURE IV-2f. Installation: New Cumberland AD

Location	AUTODIN Switch	Send	Receive
DAAS Dayton, Ohio	Gentile "A"	138,356	47,847
	Gentile "B"	167,419	49,092
	Andrews	147,742	11,448
	Ft. Detrick "A"	146,613	49,560
	Ft. Detrick "B"	159,823	55,620
	Hancock	186,086	19,609
DAAS Tracy, Calif.	Albany	170,808	14,851
	McClellan	158,395	18,994
	Norton	165,608	224,021
	Tinker	145,872	16,844

FIGURE IV-3. Defense Automatic Addressing System Office (DAASO) AUTODIN Traffic Loads for December, 1982

control then takes appropriate action on the requisition and issues a materiel rebase order through DAAS to the appropriate Depot, using AUTODIN. DAAS then forwards this requisition to the appropriate depot; throughout this process copies of status documents are routed to appropriate locations (customers, Presidio, etc.). As a result, a single order may cause many messages to be sent over AUTODIN.

The traffic generated by this system comprises about 30 percent of the total AUTODIN CONUS traffic.

Meetings were held with DLA representatives to determine what could be done to alleviate future problems.

The situation is as follows. DLA has nineteen terminals connected to AUTODIN at DLA Supply Centers/Depots and Defense Contract Administration Regions. These terminals handle narrative classified and unclassified messages and unclassified data pattern messages from telecommunications centers and unclassified data from/to DLA Supply Center/Depot Data Processing Installations. The Defense Contract Administration Regions use magnetic tape for incoming and outgoing messages to the telecommunications centers.

B. STATUS OF DAAS SYSTEMS

At one time DLA (in 1983) decided to replace the AUTODIN terminals, and a request for proposals (DLA RFP DLA 180083-R-0280) was released to obtain bids for the replacement effort. After some study and several meetings, this request for proposal was withdrawn (in June). Instead, DLA is embarking on a plan to modernize the telecommunications centers, which will include updating the procedures used as well as the equipment used. This should be cost-effective and also reduce message handling exchange time between depots and inventory control points.

Our work primarily concentrated on the AUTODIN usage with the intent of reducing this usage, but we (and DLA) have also considered the DLA Teleprocessing Network (DLANET), Fig. IV-4, and the Inter-Service/Agency Automatic Message Processing Exchange (I-S/A AMPE) plans. At the present time we are helping with the studies of the use of DDN for both the DAAS sites and the DLANET, and this looks most promising. The Services are also moving logistics traffic onto DDN and this appears overall to be a movement toward integrated, interoperable systems which should have a significant cost benefit to DoD as well.

The major actions in a plan which is now in the early stages of implementation are as follows:

1. The request for bids for a new set of AUTODIN processors for the DLA has been cancelled.
2. In the next 6 to 18 months DLA will study the offloading of AUTODIN traffic within 19 DLA sites and the DAAS. A study is now in process for an implementation and installation plan to facilitate this action.
3. A plan to use DDN for this traffic will be studied and a design formulated.
4. The largest AUTODIN users to the DAAS sites have been targeted for future offloading of traffic. DDN interfaces and DDN usage for this purpose are being studied.

C. SUMMARY

The potential overloading of AUTODIN by logistics traffic has led to actions on the part of DLA which could reduce total AUTODIN CONUS traffic by 10 percent in the near future and, with some support from the Services, by another 10 percent in a two- to three-year period.

The first action in this process was recently taken by DLA and consisted of cancelling a recently issued request for bids for new AUTODIN I processors. DLA may instead interconnect their sites using either leased lines or DDN. This would reduce the total traffic on AUTODIN by approximately 10 percent. The second step would be to connect several of the largest Service logistics users directly to the two DAAS sites using leased lines or DDN, which will reduce total AUTODIN traffic by another 10 percent. The interconnection of the DLA sites should require about one year and the connection of the Service sites another year. These steps will also allow for future expansion, both of traffic within DLA and of traffic between Service users and the DAAS sites, which are the most used AUTODIN form of logistics traffic.

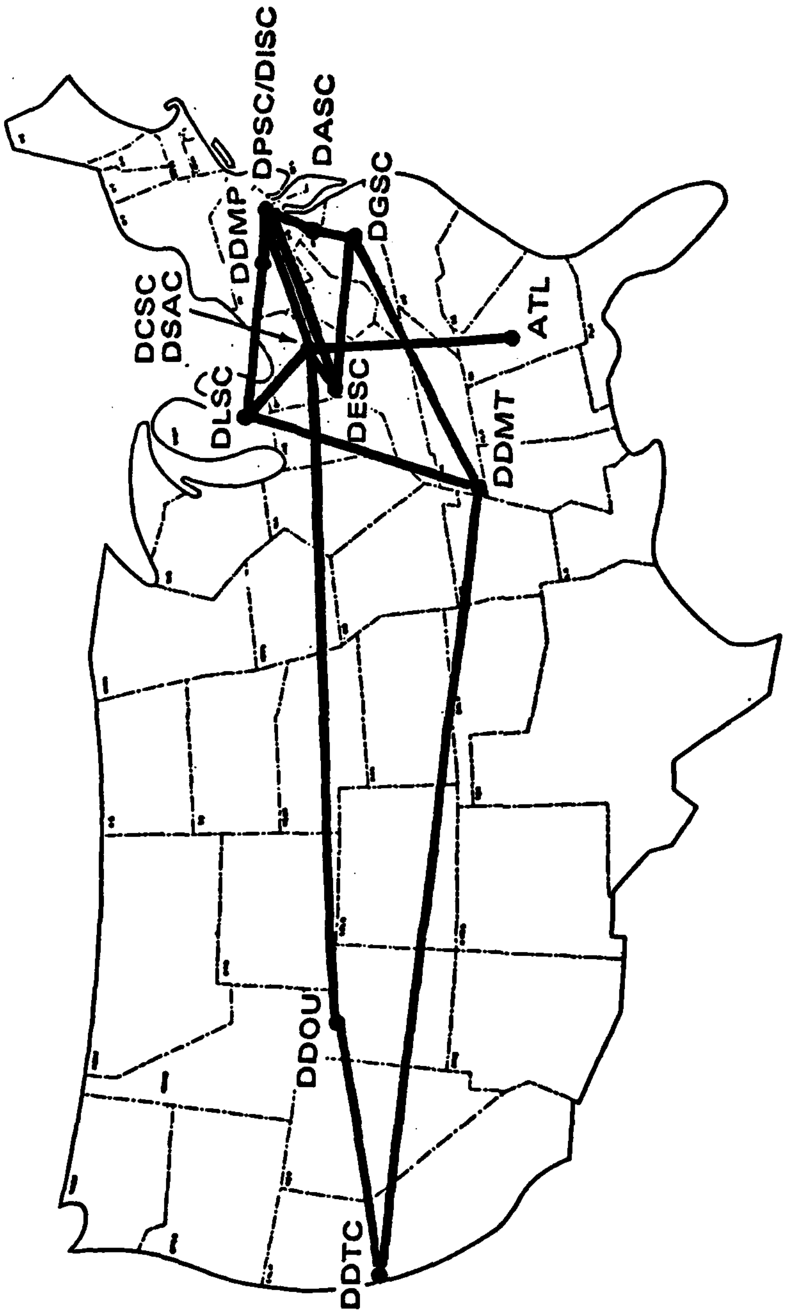


FIGURE IV-4. DLANET, December 1982

V. DATA BASES ON COMPUTER NETWORKS

A. DATA BASE PROGRAMMING LANGUAGES

1. Introduction

As observed in previous reports (Refs. 14 & 15), research and development on data bases and programming languages has proceeded more or less independently. The integration of the two areas has been somewhat haphazard and produced for practical demand rather than from any initial attempt to find a unified approach to both data bases and programming languages. In this section, we shall investigate various attempts to integrate the two, paying particular attention to data base adjuncts to Ada and to efforts to integrate data base management with some standard operating systems.

The interfaces between data bases and programming languages can be classified into a few distinct groups.

- Subroutine Interfaces

  This is the traditional method, and is to be found in the early interfaces for Fortran and data base management systems. Fortran-Codasyl interfaces are built in this way, as are certain PL/I interfaces.

- Preprocessors

  The Codasyl-Cobol interface is an example of this. The programming language is extended and a program is first fed through a preprocessor that resolves data base names and generates the appropriate subroutine calls. The preprocessor may also do some limited type-checking. A more recent example of this kind of interface is Adaplex, described below.

- Query Languages

  Most query languages are not full programming languages in that they are limited by one or more of: inefficiency, inadequacy of their data type system, lack of adequate forms of procedural abstraction, and inadequate operating system interfaces. In spite of these criticisms, a good query language can be used to generate most queries much more easily than through a programming language interface. Some query languages are being developed to the point of being adequate interactive programming languages.
- Embedded Languages

  Given that a query language is desirable, but inadequate, a common solution is to <u>embed</u> the query language in a full programming language. The query is passed through the host language (often as a character string) and the results of the query are returned as a file or as some form of query. Embedded languages are usually appropriate only for conveying a more complicated structure than a relation to the host language.
- Modified Languages with a Built-In DBMS

  A prime example of this is Pascal/R. A Pascal compiler was extended to include a data-type relation and the necessary operators on this structure. The compiler was modified substantially to include an "internal" data base management system.
- Special-Purpose Languages

  Plain and Rigel (Refs. 16 & 17) are both examples of new languages that have been developed especially for data base work. There are also a number of research efforts, Taxis and Gallileo, for example, to produce languages whose purpose is to understand the relationship between data types and data models.

- Interactive Languages

  Although not in the traditional sense of data base programming languages, the interactive environment of these languages is such as to make them quite usable for (small) data base work. The idea of a work space is a particularly simple method of dealing with "persistent" data.

2. <u>Data Base Extensions for Ada</u>

Although Ada was developed originally as a language for "embedded systems," its type system and control structures, both of which are in the tradition of Pascal-like languages, make it as suitable as those languages normally used for data processing. Ignored in the design of Ada, data base interfaces are now seen as being highly necessary for the success of that language, especially as a DoD standard. One attempt to provide this, which we have already examined, is Adaplex. This language provides some added control structures and extends Ada's data types to handle standard record-oriented data processing problems in a fashion similar to that provided in most data base query languages. In addition, the underlying data model provides greater representational power than most existing data base management systems. Later in this review we shall argue that the augmented data types (Set and Entity) are not compatible with the other data types in Ada. Strictly speaking, they are not data types at all. This has two important drawbacks.

- Only a limited number of Ada types may be components of Adaplex's added types. These are sufficient to provide for many business-oriented data processing needs, but do not allow for many extended applications. For example, Ada arrays cannot reside in Adaplex data bases; therefore, data bases with any kind of engineering application are unlikely to be built in Adaplex.

- Since Adaplex types are not "first class" Ada types, they cannot serve as parameters for other (generic) Ada objects. This means that the Adaplex sub-language cannot be extended to handle more complicated applications. Instead, data must be mapped into some other Ada data structure before the programs can be written.

What other approaches are there to the problem of adding a data base component to Ada without further increasing its complexity or violating its type system? Before answering this we should observe that any data base management system requires at lease one "bulk" data type such as relation, record class and set (Codasyl), set (Adaplex), etc. The only bulk type provided by Ada is ARRAY, and this (on its own) will not be appropriate for data base work. The crux of the problem is therefore to provide a bulk type (a) which can be simply combined with other types in the language and (b) whose objects can be made persistent. This last point means that, like objects in the data base management systems mentioned above, they should be capable of persisting from one invocation of a program to the next and should be shareable by more than one program.

We shall briefly examine four other strategies for providing the necessary data base adjunct to Ada. They are:

1. Providing low-level support for data bases, through the appropriate packages, but leaving the data base construction to the programmer.
2. Building direct, but type-checked, interfaces to existing data base management systems.
3. Providing persistence directly to Ada data types.
4. Using an approach similar to that of Adaplex, but using a different data model.

While the problem of persistence has not been tackled for Ada, the problem of sharing is dealt with extensively. Program and structure declarations are shared through packages, while run-time sharing of information among processes is performed through tasks. We shall concentrate primarily on the exploitation of the packages for data base work, but will discuss briefly the use of tasks for handling concurrency problems.

a. Low-Level Support. At a physical level there is remarkable uniformity among data base management systems: they provide methods of indexing and methods of storing sequences or sets of uniformly typed objects. Without worrying about the higher or "conceptual" issues involved, let us look at how one might provide persistent versions of the appropriate basic data structures within the language.

Viewing an index as an abstract data type, the basic operations required are:

- Creation of an index
- Insertion
- Deletion
- Look-up
- Generation of all the key-value pairs stored
- (If appropriate) generation of all the key-value pairs stored whose keys lie within a given range.

The last of these is appropriate to B-tree methods of indexing but not to hashing techniques. In view of the fact that we want to be able to generate all of the key-value pairs stored, we may use the same technique to store a sequence simply by having the key consist of the objects we want to store and the associated value a "unit" object, whose value is of no importance. This would provide a (possibly inefficient) method of storing both sets and sequences depending on whether key values are constrained to be unique.

A typical package declaration for indexing structures might be:

```
generic

     type VALUE_TYPE is private;
     type KEY_TYPE is private:
     with function "<"(Kl, K2: KEY_TYPE) return BOOLEAN:
                                    --required for range searches
     UNIQUE_KEY: BOOLEAN;

package INDEX is

     type CURSOR is private;
     procedure INSERT
              (in K: KEY_TYPE: in V: VALUE_TYPE;
               out SUCCESS: BOOLEAN);
     procedure DELETE
              (in K: KEY_TYPE; out SUCCESS: BOOLEAN);
     procedure LOOK_UP
              (in K: KEY_TYPE; out V: VALUE_TYPE
               out C: CURSOR_TYPE; out SUCCESS: BOOLEAN);
     procedure NEXT_LOOK_UP
              (in K: KEY_TYPE; in out C: CURSOR_TYPE;
               out V: VALUE_TYPE; out SUCCESS: BOOLEAN);
     procedure FIND_FIRST
              (out K: KEY_TYPE: out V: VALUE_TYPE;
               out C: CURSOR_TYPE: out SUCCESS: BOOLEAN):
     procedure FIND-NEXT
              (in out C: CURSOR_TYPE; out K: KEY_TYPE;
               out V: VALUE_TYPE; out SUCCESS: BOOLEAN);

private
     type CURSOR_TYPE is...

end INDEX;
```

There are many ways this declaration could be simplified and more in which it could be made more complicated. Some of the reasons for this choice are given here.

1. The types KEY_TYPE and VALUE_TYPE are parameters of the generic package; an example of KEY_TYPE would be SS_TYPE (the declared type for social security number) and EMPLOYEE. UNIQUE_KEY indicates whether keys are to be unique.
2. CURSORs are needed for traversing sequences. They will be needed in range traversals, traversal of the whole table, or traversals of all records with the same key in cases in which the key is not unique. We could associate a single cursor with the package (rather than with the individual procedures) and the various procedures could work by side-effecting this variable. This causes confusion when more than one sub-programs are interleaved and each requires access to the table (Ref. 15.)
3. LOOK_UP and NEXT_LOOK_UP are both needed when there is more than one value for a given key.
4. A boolean value SUCCESS is used to indicate the success or failure of each operation. An alternative approach for LOOK_UP would be to have an access type (access VALUE_TYPE) returned. This would be NIL if the LOOK_UP failed. Unfortunately, this approach will require the package or the Ada run-times to support the appropriate storage management.

Given a particular KEY_TYPE and VALUE_TYPE, this package may be generated and used as a separate compilation units. This may now be used by a number of "applications" programs in much the same way that a data base management system is used. Note that we are assuming that there is exactly one "data base" for each package generated, and that we are assuming that the data associated with that package are persistent.

If an instance of such a package is to be a separable, persistent, object, there are certain restrictions that must be placed on what KEY_TYPE and VALUE_TYPE may be. There must be no direct references (through access types, say) to other objects available to the program. Thus, any type constructed from the basic Ada types and using records or arrays is allowable; however, pointers cannot be used, nor can task types. In addition, equality (at least) must be computable for KEY_TYPE values. Types such as reals (for which equality is undecidable) cannot be allowed.

A more general point is that a real data base implementation may require the instantiation of, say, 100 such packages. These are independent of one another and the problem of sharing data must be approached. Should data in two such packages need to be related, the only solution is to have explicit (atomic) component data types exactly in the same fashion as relational data bases use explicit, printable, types to indicate related data. In fact, mapping from higher-order data models to first normal form relations often requires the introduction of such fields. The ostensible independence from run-time, dynamic structures makes it possible to treat each component as a separable, copyable unit.

Some of these problems are mentioned under more general persistence techniques. Note that we have not dealt with concurrent access to such a data structure.

It is obvious that a generic persistent set type could be built subject to the same constraints on sharing.

b. Ada Interfaces for Existing DBMSs. This is probably the most important need for the environments in which Ada will be first used. It will be impossible to restructure data bases and entire suites of applications programs in order to mesh with the (as yet non-existent) data base requirements for Ada. The best that can be hoped for is that reasonably simple interfaces can be built between Ada and existing data bases and that these will be more easily rewritten or restructured than existing programs when the data bases themselves are ultimately restructured.

There are two general approaches to this problem. Both are implementable without the need for major additions to the Ada language and without complex preprocessors to implement these additiions.

1. Embedded Languages. The idea here is to take an existing data base query language (SQL for example) and have within Ada a processor that can call upon this language. The query is handed to the query language as a processor and what is returned is a mechanism for bringing the results of this query back into the host language (Ada). The result is usually a sequence of some sort and can be traversed using the cursor mechanism described in the previous sub-section.

   The only open question is how the returned results are to mesh with the type system of Ada. A trivial solution is to send the results (usually records) back as character strings--perhaps with format information. A better approach is to have them returned as Ada records so that the programs may be properly type-checked. For this, some sort of automatic generation program for type declarations is required. Note, however, that we run into a predictable problem with dynamic versus static type number of data types: by changing the query (the contents of a character string) one can change the type of the result. This is foreign to the principles of strong typing.

   Embedded systems are most appropriate in dealing with systems that always return "flat" results--e.g., the relational systems.

2. Direct Interfaces. These are appropriate when the underlying data base has a low-level data manipulation language available that deals with individual records or references rather than bulk entities such as relations. Typically, such an interface is provided for Codasyl systems (although the embedded language approach could

also be used.) Again, the problem is that of making the data base types agree with appropriate Ada types. For Codasyl at least, the problem is simpler in that the (record and set) types are fixed in advance and it is possible (Ref. 18) to use Codasyl Data Definition Language to generate automatically an appropriate set of Ada type declarations and procedures. This is a considerable improvement over most existing Codasyl programming environments in which data base queries are <u>not</u> type-safe and run-time type violations with disastrous consequences are common.

c. <u>Building Persistence into Ada</u>. The idea that most data base work can be accomplished by having programming languages with persistent objects (i.e., objects that persist from one invocation of a program to the next) is relatively new (Ref. 19). In Ada, for example, packages may be shared by several programs--i.e., code and type declarations are shareable and persist, but the objects themselves are associated with a particular invocation of the program. Ada itself has no persistence; however, the packages for i/o and the generic package above did provide persistent stores by side-effecting a structure that was "foreign" to Ada's run-time store.

The question is, therefore, why any Ada object cannot be tagged as persistent? The problem is with objects that refer to other objects. We cannot allow a persistent object to refer to a run-time object, for the reference will not remain valid in the next invocation of the program. One solution is only to allow a subset of types to give rise to persistence. The restrictions are much the same as those that would normally be placed on the contents of Pascal files. Another solution is to partition the store into components that can be tagged (could it be done automatically?) to indicate in which component they are to reside.

Research on persistence is relatively recent and we do not suggest that one can count upon availability of a "persistent Ada" in the near future. Ultimately, however, persistent languages will provide better solutions to data base problems such as these.

d. Other Extensions to Ada. Adaplex presents one solution to an integrated data base environment to Ada. Are there others? We note that Pascal-R has provided a successful extension to Pascal for the relational data model. To do the same for Ada would involve restructuring the Ada compiler and extending the language. It may be possible to adopt the same approach as was taken with Adaplex and provide a preprocessor that translates the language extensions. It is an open question as to whether these approaches will be sufficiently simple and efficient to be acceptable in the Ada programming environment.

e. Summary. Of the four approaches listed above, the simplest and most likely to find immediate use are the first two: construction of data base support packages in Ada and building interfaces directly between Ada and existing data base management systems. The second two approaches, while they may provide better long-term solutions, should not be counted upon in the near future to provide reliable data base support. The problem of providing interfaces to existing data base systems will prove most important in the short term.

An area that we have yet to investigate in the context of Ada is concurrent access to data bases. It is possible that Ada's mechanisms (tasks) for concurrent programming could also be exploited for this. However, the concurrency mechanism in Ada is based upon message passing, while that used in most data base systems is based upon shared storage. The examination of the relationship will involve a consideration of what role operating systems should play in concurrency control.

3. Examples of Problems in Integrating Data Bases with Programming Languages

In order to look at other languages and in order to illustrate some of the problems associated with integrating data bases and programming languages, we shall work through some examples. We shall investigate the same problem cast in a number of different languages, both in order to understand the type or schema declarations and in order to see how the control differs in the various languages. For the most part these languages are relatively new or are experimental. They are indicative of the research that is taking place in this area.

The problems we shall look at are derived from a standard bill-of-materials scenario. In a manufacturing plant, parts are assembled from sub-parts and those sub-parts in turn have sub-parts. Note that not only does each part have a set of sub-parts, but each type of sub-part may also be a sub-part of several different super-parts. Thus the sub-super relationship is a many-many relationship on parts, as seen in Fig. V-1.

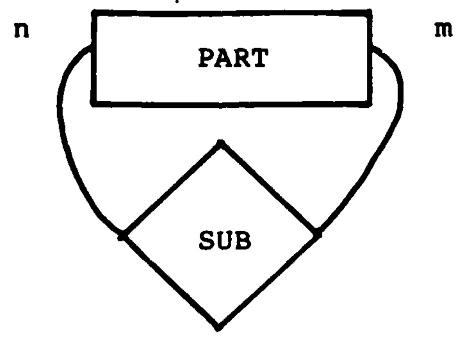


FIGURE V-1. E-R schema for part data base

Associated with this relationship may be a number, i.e., the number of sub-parts of a given type that are involved in the manufacture of a super-part of a given type. To capture this information, a different representation is needed, both of these being entity-relation diagrams (see Fig. V-2):

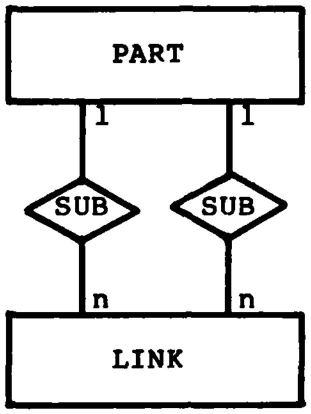


FIGURE V-2. E-R schema for part-link data base

a. Daplex. The language Daplex (Ref.20) is not a full programming language. It is designed to capture a large number of data base applications and, when it fails to do this, it is supposed to be used as a "sublanguage" of a full programming language. It is the basis of the Ada extension "Adaplex" that we shall examine below.

Daplex exploits the functional data model (Ref. 14). The term "part" is overloaded to refer both to an entity type (conventionally a record type) and to refer to the niladic function that generates all objects in the data base off that entity type. The first line of Fig. V-3 declares this. Subsequent lines declare that name, color, and weight are (single-valued) functions from this entity type to the appropriate atomic type (STRING or INTEGER).

The declaration of sub and super as multi-valued functions (i.e., functions whose value is a set of objects) from part to part follows in the next two lines. The declaration of super constrains it to be the inverse function of sub. Thus, any update to the function sub is reflected in a change to the function super.

This declaration as it stands is a gross simplification of a real schema. It is, however, the one we shall use for our examples. We have noted that a more faithful representation of the data base would include information about the quantity of each sub-part that is needed for the manufacture of a given part. In this case, the declarations would look like the standard Codasyl recursive structure; an intermediate "link" entity would have to be introduced (see Figs. V-3 and V-4).

```
declare part ( )         ->>     entity
declare name(part)       ->      string

declare color(part)      ->      string
declare weight(part)     ->      integer

declare sub(part)        ->>     part
declare super(part)      ->>     link = inverse of sub
```

FIGURE V-3. A representation of the manufacturing data base in Daplex

```
declare part ( )         ->>     entity

declare link ( )         ->>     entity

declare name(part)       ->      string
declare color(part)      ->      string

declare weight(part)     ->      integer

declare subl(link)       ->      part
declare supl(link)       ->      part

declare qty(link)        ->      integer

declare sub(part)        ->>     link = inverse of subl
declare super(part)      ->>     link = inverse of supl
```

FIGURE V-4. An alternative representation of the manufacturing data base in Daplex

The programs we shall deal with will, however, exploit the first schema declarations. The first of these is a simple traversal of the data base to print out the names and weights of the red parts in the data base (see Fig. V-5).

```
for each p in part such that color(p) = "RED"
  do {print (name(p)); print (weight(p))}
```

FIGURE V-5. Retrieval of all the red parts, coded in Daplex

In Daplex this resembles a standard iteration in a programming language; the only difference is that name and weight are treated as functions rather than record selectors.

The second, and more complicated problem, is to print out the names of all sub-parts of a given part. Daplex includes a transistive closure operator that makes this particularly easy. (It has been recognized that this is missing in most of the relational query languages.)

```
define allsubs(part)   ->>    transitive of sub(part)

   for the part such that name (part) = "WIDGET" do
            for each p in allsubs(part) do
          {print (name(p)); print (color(p))}
```

FIGURE V-6. A query in Daplex to print the subparts of a widget

The first line of Fig. V-6 contains a definition of allsubs as a transitive closure of the function sub (with reference to the first schema). The second line contains a "FOR THE" construct. This binds the (variable) part to a part entity provided exactly one such part exists satisfying the subsequent condition. Note the further overloading of the term part. With these definitions the query proceeds as before.

In this example, the two programs were remarkably simple. It should be observed that had we wanted to print out not only the sub-parts but also the quantities of each sub-part involved in the manufacture of a given part, we would have had to use the second schema. In this case, the sub-part query could not have been so simply stated in the language. In fact, it is possible that one would have to resort to an external or "host" language in order to accomplish it.

In the subsequent examples, the sub-part query will not be as simple. We will have to resort to explicit breadth-first search in order to implement it. In this, a list (called LEVEL) of current parts is kept. At each iteration, the set of all sub-parts of parts in LEVEL is found. The parts on LEVEL are added to an accumulator, ALLSUBS, and NEXTLEVEL is assigned to LEVEL. The process is repeated until LEVEL is empty, i.e., all parts are "terminal", i.e., not part of this manufacturing operation. Also note that should the data base contain a cycle, this and subsequent programs will not terminate.

b. Pascal/R. Daplex was chosen as the first language because of its relative simplicity and resemblance to a relational query language even though the underlying model was not, in fact, relational. We now turn to a proper relational system. Since the sub-parts query cannot be performed in most relational query languages, we shall examine Pascal/R, the relational data base extension to Pascal (Ref. 21). Figure V-7 shows the type of declarations for this data base. Note the introduction of a new type relation which is parameterized by record type and key fields. Normalization demands that we create two relations: PartRel and LinkRel to represent the data base. The relation LinkRel can contain a quantity field, and this declaration corresponds most closely to the second of the Daplex declarations. The data base declaration is a special form of record declaration that only allows relations as components.

```
type
   Pnum          = integer;
   PartRec       = record
                           pn:Pnum;
                           name:string;
                           weight:integer;
                           color:(RED,GREEN,BLUE)
                   end;

   LinkRec       = record
                           super, sub:
                           Pnum;
                           qty: integer
                   end;

   PartRel       = relation <pn> of PartRec;
   LinkRel       = relation ,super, sub> of LinkRec;

var
   Manuf         : data base
                           parts: PartRel;
                           links: LinkRel
                   end;
```

FIGURE V-7. Pascal/R data structure declaration for manufacturing data base

That the data base declaration is special is illustrated in Fig. V-8. It is used as a variable declaration in much the same way as a File is used in regular Pascal and is another form of persistent data. There are, for implementation reasons, restrictions on what can be placed in a relation. Arrays are not allowed as domains nor, obviously, are pointers, files, or other relations.

The code for the program is straightforward; it is illustrated in Fig. V-8. The foreach loop allows traversal of a relation and is an addition to Pascal's repertoire of loop control forms.

```
program Redparts( Parts, Output );
type
   Pnum     = integer;
   PartRec  = record
                    pn:Pnum;
                    name:string;
                    weight:integer;
                    color:(RED,GREEN,BLUE)
              end;
   LinkRec  = record
                    super, sub:
                    Pnum;
                    qty: integer
              end;
   PartRel  = relation <pn> of PartRec;
   LinkRel  = relation <super, sub> of LinkRec;
var
   Manuf    : data base
                    parts: PartRel;
                    links: LinkRel
              end;
begin
   with Manuf do
      foreach p in parts do
         if p.color = RED then
            Write( p.name, p.weight)
end.
```

FIGURE V-8. Pascal/R program to print the red parts

Figure V-9 illustrates breadth first traversal of the data base. The main variable declaration contains some temporary (i.e., nonpersistent) relation declarations. Below this we see a procedure that has relations as parameters. The purpose of this procedure is to compute the next level of sub-parts in the data base. The computation of NewLevel is essentially a double join. The notation [each <rec> in <rel>: <pred>] is a notation for a restriction, the set of <rec> in relation <rel> that satisfy the predicate <pred>. The notation Some <rec> in <rel>: <pred> is true if and only if some tuple (<rec>) in the relation <rel> satisfies <pred>.

The main body of the program first computes the relation consisting of all parts whose name is 'WIDGET'. This is used as the first level. Calls to NextLevel compute the set (relation) of parts at subsequent levels. At the same time, each level is accumulated in Allsubs. Finally, interesting domains of this accumulated relation are printed out. Note the use of :+ and := as operators on relations. Pascal/R has a number of such operators, and many queries can be executed concisely using them. They can also be performed through use of explicit controls, such as foreach loops. It is quite common to find that Pascal/R provides several ways of implementing a given query. In general, the more concisely it is implemented, the more chance there is of its being properly optimized.

c. Adaplex. There are certain similarities between Pascal/R and Adaplex (Ref. 22). The former is an embedding of a relational data type in Pascal; the latter embeds a functional data model in Ada. Adaplex owes its extended type declarations to Daplex, examined above. The type PART (Fig. V-10) is declared as an entity type with various fields. Within entity declarations certain additional field "types" are possible. Name, Pnum, and Color are all treated as functions on this type. Sub and Super are multivalued functions that map a part into sets of parts.

```
The Unique program Components(People, Output );
...    {Type declarations as in Figure V-8}
var Temp, Level, Allsubs:PartRel; p:PartRec;

procedure NextLevel(OldLevel: PartRel; var NewLevel: PartRel);
{given a set of parts in OldLevel, it calculates the next
subparts in NewLevel}
begin
  with Manuf do
    NewLevel := [each q in Parts:
                   Some b in links:
                     (q.pn = b.sub and
                      Some p in OldLevel: p.pn = b.super) ];
end;

begin
  with Manuf do
    begin
      Allsubs := []; Level := [each p in Part: p.name='WIDGET'];
      while not empty( Level ) do
        begin
          NextLevel( Level, Temp ); Allsubs :+ Temp; Level :=
          Temp end;
          writeln( 'The subparts of a Widget are : ');
          foreach p in all subs do
            Writeln( p.name, p.weight)
        end
end.
```

FIGURE V-9. A Pascal/R program to print the subparts of a Widget

assertion makes Pnum a (logical) key for PART. The difference between entities and records--apart from set fields--is that with each entity type is associated a persistent extent. Thus, the term PART defines both a data type and a set of objects of that type (much as in Daplex).

This simple data declaration does not do justice to the extended data model used in Adaplex. In particular it is possible to make certain entities sub-types of other entities and to constrain the way in which these sub-types overlap. For example, it is possible to define an entity type PERSON with sub-types STUDENT and EMPLOYEE. The sub-types have all the attributes (functions) defined on PERSON and some additional ones as well. It is also possible to state whether STUDENT and EMPLOYEE entities overlap (note the comparison with variant records).

```
data base Manuf is
   type COLORTYPE is (RED,GREEN,BLUE);
   type PART;              --Needed for recursive definition

   type PART is entity
      Name : STRING(1..20);
      Pnum   : INTEGER;
      Color : COLORTYPE;
      Sub   : set of PART;
      Super  : set of PART;
      Qty: INTEGER;
   end entity;
   unique Pnum within PART;
end Manuf
```

FIGURE V-10. The data definition in Adaplex for the manufacturing data base

The code in Fig. V-11 is almost self-explanatory and bears several resemblances to Pascal/R and Daplex. The use of with and use follows the syntax for packages. The use of by is a convenient way of expressing the need for sorted output--something that is lacking in Pascal/R and many query languages. The atomic ... end atomic construct specifies a transaction--a section of code that must run, or appear to run as an indivisible operation.

```
with Manuf;
use Manuf;
Print_Widgets: atomic
   for each P in PART where color(P) = RED by Name(P)
   loop
      PUT(Name(P)); PUT(Weight(P)); NEW_LINE;
   end loop;
end atomic;
```

FIGURE V-11. Adaplex program to list the red parts

The use of a separate, nonpersistent set type is illustrated in the sub-parts query of Fig. V-12. Set types are initialized, automatically, to the empty set. The construction {<ent> in <set> where <pred>} follows that of Pascal/R. include ... into and exclude ... from are side-effecting union and difference operators. The difference operator can be used to empty a set. Level is defined as a set of PART entities and Sub is a function defined on a PART entity. The use of Sub(Level) is an overloading of Sub to work on a set of entities. This is a common and convenient abbreviation for what would otherwise have to be specified by an explicit mapping or control form.

```
with Manuf;
use Manuf;
Subparts:
  declare
    Level, NextLevel, Allsubs : set of PART;   -- Initialized
                                                  to empty
  atomic
  include /P in PART
      where Name (P) = "Widget"/ into Level;
    while Level is not empty loop
      include Level into Allsubs;
      include Sub(Level) into NextLevel;
      exclude Level from Level;               -- empty the set;
      include NextLevel into Level;
      exclude NextLevel from NextLevel;       -- ditto
    end loop;
    for each P in Allsubs loop
      PUT(Name(P)); PUT(Weight(P)); NEW_LINE;
    end loop;
  end atomic;
```

FIGURE V-12. Adaplex program to print the sub-parts

Pascal/R and Adaplex have a number of similarities in design. While Pascal/R is a straight extension to Pascal and is implemented by modifying the compiler, Adaplex is pre-processed Ada. It is perhaps unfortunate that the Ada compiler could not be directly extended. Sets, for example, are not really "first class" types. They cannot be freely mixed with other types. Thus, one cannot have set of array ... or array ... of set of ... . This is particularly unfortunate in a language that does allow the use implementation of generic types (to which set types should conform).

d. Pascal-Codasyl. Lest is should be thought that these examples are not an improvement of the status quo, let us briefly examine a more conventional approach to the problem. The following is a Pascal implementation of the queries against a Codasyl data base. The code presented here is, in fact, working code based on a Pascal-Codasyl interface (Ref. 18). The advantage of this interface is that Codasyl records are presented as Pascal types and the Pascal type-checker makes sure that the programs are at least type-correct. In the case of the sub-parts problem, type errors are the most common form of coding error.

The Codasyl schema is straightforward and warrants little description. There are two record classes, Part and Link, and two sets, Sub and Sup, that link these classes in an obvious way. In addition, Part_ref and Link_ref are reference types for these record types and are used explicitly as Codasyl currency pointers.

In Fig. V-13 FINDFA_PART is a procedure (provided by the interface) to find the first part in the (default) area. FINDNA_PART is a find-next-in-area and differs from normal implementations in that it takes the previous currency as an explicit reference. GET_PART instantiates the record from its currency. Apart from this, the code is standard Pascal and should present no difficulty.

The problems of implementing the sub-parts query are considerably greater. Pascal has no built-in set manipulation operators.* We must therefore develop these ourselves. In a standard data-processing environment--using Cobol, for example--these might be implemented using sort-merge techniques in secondary storage. Pascal does not have generic types, so that if we wanted to manipulate two distinct set types, we would

*It does, but Pascal sets are not general and will not do for this application.

have to duplicate our code for the different types. The initial declarations of Fig. V-13 are concerned with this. INSERT, REMOVE, and CHOOSE will be adequate for our application. In general, other set operations such as UNION and DIFFERENCE would be desirable.

```
var Pref: Part_ref; P: Part;
begin
  OPEN(MANUF);
  FINDFA_PART(Pref);
  while PREF FOUND do
    begin
      GET_PART(Pref,P);
      if P.COLOR = RED then WRITELN(P.Pname, P.Weight);
      FINDNA_PART(Pref,Pref)
    end
  CLOSE(Manuf);
end;
```

FIGURE V-13. Pascal-style Codasyl query for red parts

The procedure Sub_widgets in Fig. V-15 falls into three components: the first to find the part(s) named 'WIDGET', the second to find the set of sub_parts, and the third to print out fields from this. The first part proceeds similarly to the previous example, and sets LEVEL to be the set of parts named 'WIDGET'. The outer loop of the second part recomputes successive levels of parts. In the inner loop, we first choose a part from the current level. We then perform a down-and-up traversal of the Link record class. These procedures draw upon additional set manipulation procedures shown in Fig. V-14.

- FINDFS_SUP(PREF,LINKREF) finds the first link reference in the SUP set owned by PREF.
- FINDOS_SUB(LINKREF, PREF1) finds the part-reference owner of the SUB set of which LINKREF is a member.

- FINDNS_SUB(LINKREF, LINKREF) finds the next member in the SUP set. Note that, like FINDNA_PART, the previous currency is a parameter.

Finally, all the sub-parts are accumulated in ALLSUBS. We may then use our own set manipulation procedures to traverse this set and print the fields of interest.

```
typePart_set ...  {type declarations for sets of parts}

VAR Emptyset: Part_set

procedure INSERT(PREF:Part_ref; var PS: Part_set);
...

procedure REMOVE(PREF:Part_ref; var PS: Part_set);
...

procedure CHOOSE(PS: Part_set; var PREF: Part_ref);
...

{etc. and other set manipulation procedures}
```

FIGURE V-14. Auxiliary procedures needed for sub-parts query

4. Conclusions

These examples have been chosen to illustrate some of the ideas that have been applied to the improvement of interfaces between data bases and programming languages. The last example (which is shorter and more transparent than conventional code for the same purpose) illustrates, perhaps, why an improvement is needed. There is, of course, endless discussion on what control structures and what syntax are most appropriate for such an interface.

The more serious problems, and they are still research problems, are concerned with data types. In each case cited above, the inadequacies lay in the generality of the type system of either the language or the embedded data model. A strongly typed interface is essential. Most errors in complex data base code are type errors; and in simple (query) languages that are not type-checked, type errors result in baffling run-time messages for the end-user.

```
procedure Sub_widgets;
var P: Part; PREF, PREF1: Part_ref; LREF: Link_ref;
        LEVEL, NEXTLEVEL, ALLSUBS: Part_set;

begin
  OPEN(Manuf)
  LEVEL:=Emptyset;
  FINDFA_PART(PREF);
  while PREF.FOUND do
    begin
      GET_PART(PREF,P);
      if P.Pname = 'WIDGET' then INSERT(PREF, LEVEL);
      FINDNA_PART(PREF,PREF)
    end

  ALLSUBS:=EMPTYSET;
  while NOT(EMPTY(LEVEL)) do
    begin
      NEXTLEVEL:=Emptyset
      while NOT(EMPTYSET(LEVEL)) do
        begin
          CHOOSE(LEVEL,PREF); REMOVE(PREF,LEVEL);
          FINDFS__SUP(PREF,LINKREF);
          while LINKREF.FOUND do
            begin
              FINDOS_SUB(LINKREF,PREF1);
              INSERT(PREF1,NEXTLEVEL);
              INSERT(PREF1, ALLSUBS);
              FINDNS_SUP(LINKREF,LINKREF);
            end;
        end;
      LEVEL:=NEXTLEVEL
  end;

  while NOT EMPTYSET(ALLSUBS) do
    begin
      CHOOSE(ALLSUBS,PREF); REMOVE(PREF,ALLSUBS);
      GET_PART(PREF,P);
      WRITELN(P.Pname, P.Weight)
    end
end;
```

FIGURE V-15. Pascal-type Codasyl query for all sub-parts of a widget

Future attempts at designing programming language--data base interface should be judged by the generality and completeness of the type system. It is to be hoped that no one in the future will design a language that does not support a data base or a data base management system that is not supported by a language.

B. DATA BASES AND EXPERT SYSTEMS

1. Introduction

The last few years have seen an increasing commercialization of Artificial Intelligence. The so-called expert systems that were, for many years, experimental tools or at best programs that could be of assistance to a few highly-trained people are now being advertised as general-purpose tools that can be used in any area that requires substantial problem-solving capabilities, together with a knowledge base.

To what extent are these claims justified and what will be the impact, if any, on C^3I opertations? If a substantial knowledge base is to be built, will it be possible to use existing C^3 data bases and applications software or will the data bases have to be completely restructured? In this section we shall take an exploratory look at some of these problems and examine some of the tools that may be appropriate for integrating data bases and expert systems.

In spite of all the publicity that has been given to expert systems in the past year or two, there is still some argument as to whether they will prove valuable. Of the various systems cited here, few to our knowledge are yet in commercial production, and whether or not the substantial costs that have been incurred in developing them have yet been justified is still open to debate. On the other hand, one or two smaller-scale systems--computer configurers are an excellent example--are in day-to-day use and have almost certainly justified their investment.

Moreover, a large number of so-called "decision support systems" has been implemented with spectacular results; and the fact that expert (as opposed to decision support) systems have not yet borne fruit may be more a matter of classification than of substance. Before looking at individual systems, a few words on the classification of systems are in order.

The first observation is that expert systems are, for the most part, used by experts. In the examples cited, and in all the others we have seen, the purpose of the system is to capture a body of knowledge and to implement complex reasoning or search mechanisms that could be understood by the user. In this sense an expert system is no different than any other form of computer aid: it serves to amplify the abilities of an expert in some area by representing a larger knowledge base and by carrying out computations more efficiently and accurately than the user can. In the case of expert systems, the knowledge base may be encoded in a set of rules or in some semantic network, and the computations are often some form of inferencing that is done on the basis of those rules or the network.

It is only the flavor of the representation and of the computations that distinguish expert systems from decision support systems. The latter usually combine access to data base management systems with a set of readily accessible tools for the extraction and manipulation of data. Again, a decision support system is only useful to an expert, i.e., one who is technically knowledgeable in the domain of interest. If there is any difference in the use of decision support systems and expert systems it is that in the former, the computation steps required to achieve a result are well understood by the user. In an expert system, the user will understand the principles on which the computation proceeds but may not know how a specific result was achieved. For this reason an important part of the interaction with an expert system is some form of explanation. It has (surprisingly recently) been realized that, since the users of expert systems are themselves experts, some form of explanation of what intermediate hypotheses the system investigated in order to obtain a given result is needed if the system is to be credible and flexible enough for interactive testing of alternative hypotheses.

2. Some Well-Known Systems

The following review of some of the frequently cited systems is taken largely from direct quotations from relevant papers. They are included here simply for convenience and also in an effort to answer the basic questions:

- What does the system do?
- Who uses it?
- How widely is it used?
- What language or other tools were used to build it?

3. The INTERNIST System

Internist is a system that aids physicians, usually internists, in diagnosis of Internal Medicine disorders. It was developed as a result of extensive study of how physicians themselves perform this diagnosis and has proved to be "effective in sorting out the complexities and rendering a correct diagnosis in the great majority of clinical cases tested." The implementation language is INTERLISP together with external files that serve as data bases. The use of INTERLISP's "spaghetti stack" was found to be extremely useful for context management and backtracking. It is claimed to have an extremely friendly user interface.

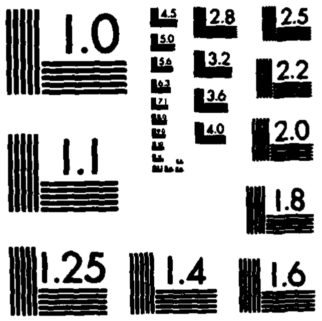
As for the efficacy of the system, the following quote is taken from Ref. 23:

> It became clear on the basis of extensive testing that many aspects of INTERIST's performance could be significantly enhanced if it were possible to deal with the various component problems and their interrelationships simultaneously. In many cases, especially those requiring diagnostic consultation, patients can present

a number of concurrent clinical problems. While certain patterns of co-occurrence of disease are more likely than others, one cannot exclude the possibility of encountering a dozen or more hitherto unrelated disease entities in any given patient. A clinician seeks a unitary cause, if possible, to account for all observed findings in a patient. This does not necessarily call for the identification of a single disease entity. A unitary hypothesis can include a number of distinct entities, which are interrelated via some pathophysiological (i.e., causal) mechanism. Our first approach to computer-based diagnosis (which predates INTERNIST-I) was oriented toward the identification of unitary hypotheses. The main problem with this precursor system was its inability to deal with real-world clinical problems. The analysis of actual patient data (as opposed to test cases) frequently led to the wrong conclusion, even in relatively uncomplicated clinical problems. The clinician, unlike the program, is not misled by a compulsion to search out unitary hypotheses that can explain all the data. The clinician exercises judgement in disregarding major portions of the clinical findings of a case, while focusing on the problem and/or problems implicit in a subset of the data. In an effort to emulate the skilled clinician's capability to partition a clinical problem into 'obvious' sub-problems that could be considered separately or together at will, we developed a problem-formation and attention-focusing heuristic and incorporated this in the system called INTERNIST-I. Though exceedingly robust and proven effective in solving a wide range of difficult clinical problems, the sequential approach to problem formation and solution incorporated in the INTERNIST-I is not without

AD-A144 271 C3I (COMMAND CONTROL COMMUNICATIONS AND INTELLIGENCE) 2/2
INFORMATION SYSTEMS..(U) INSTITUTE FOR DEFENSE ANALYSES
ALEXANDRIA VA T C BARTEE ET AL. APR 84 IDA-P-1774
UNCLASSIFIED IDA/HQ-84-28529 MDA903-79-C-0018 F/G 9/2 NL

END


1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

shortcomings. Reactions of clinicians who have interacted with the system over the past two years have pointed to a dissatisfaction with the tendency of the program, in complex cases, to begin its analysis by considering wholly inappropriate problems, on which it may spend an inordinate amount of time. This rarely leads to a false conclusion but it does prolong the sessions of terminal interaction unnecessarily. The reasons for this phenomenon relate the system's inability to perceive the multiplicity of problems in a case all at once. The design of INTERNIST-I was motivated by the need to formulate and focus attention on individual components of complex clinical problems.

Experience with this system suggests, however, that a multi-problem focus and prior attention to the interrelationships among hypothesized disease entities might yield patterns of behavior that would appear more reasonable and acceptable to the clinician users of the system. This has led to the development if INTERNIST-II, a system embodying strategies of concurrent problem-formation. INTERNIST-II employs a hypothesis generator that uses the concept of constrictor to delineate the top-level structure of a complex problem, and a modified scoring algorithm that considers within each sub-problem only those findings judged to be relevant in that context. As of this writing, the heuristic multi-problem formation and reduction procedures have been implemented and tested on a large number of cases with resulting initial problem formulation that surpasses INTERNIST-I performance in all complex cases. The synthesis procedure has been incorporated in the system only recently, however, and is not currently being exploited to the full extent anticipated. The intention is to invoke the synthesis heuristic during the course of state-space expansion in order to bias sub-problem selection toward potentially unified constructs. At present, the procedure is invoked only in ex-post-facto fashion after all conjunctive problems have been reduced to simpler form. The system is incomplete in other respects as well; it has not yet been fitted with the type of inquiry capability used in INTERNIST-I to test and evaluate hypotheses. However, we expect that the improved problem focus of INTERNIST-II can be counted on to provide more discriminating information-gathering capability than was possible in the predecessor system. What remains to be seen is the adequacy of the state-space formalism as a framework within which appropriate reformation of the problem-set can be recognized and/or developed in those cases when the initial focus is, at least in part, incorrect. One of the great strengths of INTERNIST-I is its ability to shift the focus of attention from one problem to another on the basis of newly-derived data. To attain comparable facility in the multiple problem context of INTERNIST-II will require more elaborate control strategies, because new evidence will be expected in most cases to call for alteration of only one component of the total problem structure. Quite radical restructuring of the total problem focus may be called for in some cases. Special control strategies will also have to be devised for a search mechanism that calls for the conclusion of the primary form of a disease only after other etiologies have been excluded. Similar search strategies, involving directed investigation of the INTERNIST-II state space, will need to be incorporated in the system in order to deal with this class of decision problem. There is also a clearly perceived need for extending the dimensionality of the hierarchy of disease categories so that additional relationships, not now capable of representation, may be made available to the diagnostic programs. This will achieve the full power of the constructor-based multi-problem generator.

Much of the machinery needed to carry out critical experiments with respect to the issues discussed above is already in place, and additional results may be expected in the near future.

4. MYCIN

MYCIN is again a diagnostic system designed to aid physicians. Unlike INTERNIST, the domain of interest is somewhat smaller, being limited to diagnosis and the treatment of bacteremia (bacteria in the blood) and meningitis (bacteria in the cerebrospinal fluid). Note that the program also provides aid in the selection of drugs.

Like INTERNIST, a friendly user interface is claim that involves the generation of English sentences. It is also written in INTERLISP and involves some 600 rules that deal with diagnosis and treatment. It is also under development and involves collaboration with the researchers and a specific group of physicians that form its current base of users. More recently the system has been given the ability to acquire new rules through interaction with the clinical expert. The following assessment of how well the system works is taken from (Ref. 14).

In order to assess how well the system operates, one must take into account the design criteria and the degree to which the system meets them. Among the concerns of the system developers were the following questions posed by physicians about a new system:

1. Is its performance reliable?
2. Do I need this system?
3. Is it fast and easy to use?
4. Does it help me without being dogmatic?

5. Does it justify its recommendations so that I can decide for myself what to do?
6. Does use of the system fit naturally into my daily routine?
7. Is it designed to make me feel comfortable when I use it?

We have already alluded to the fact that MYCIN is making strides in acquiring a wider knowledge data base through the relatively painless method of an expert user interface and that it does so in simple English. When the truth of a premise condition is best determined by asking, the physician enters the appropriate response, and the program continues. The knowledge expressed in MYCIN rule is seldom definitive, but tends to include "suggestive" or "strongly suggestive" evidence in favor of a given conclusion. Then, it strives not to be dogmatic. MYCIN has a clear sense of its limitations. Although the system has served us well to date, it does have several recognized inadequacies and can only be seen as a first step toward the development of a coherent theory of the management of uncertainty in complex reasoning domains. Perhaps the greatest advantage of the rules used in MYCIN is the way in which they facilitate the development of mechanisms for explaining and justifying system performance. These capabilities also contribute greatly to MYCIN's educational role. The explanation program has two options. One is limited in scope, but is fast and easy to use. This option is more powerful, but it involves language processing and it is therefore less reliable and more time-consuming. (MYCIN has a large dictionary, but questions must be kept short, without subordinate clauses. It takes novice users several sessions before they learn the best way to phrase questions so that MYCIN will interpret them properly). This question/answering capability is designed to allow full access to all system knowledge both static (i.e., facts and rules obtained from experts) and dynamic (i.e., conclusions reached by the program for the specific consultation session). An evaluation of MYCIN's decisionmaking capabilities leaves many points unclear. First, any evaluation is difficult because there is so much difference of opinion in this domain, even among experts. Actual clinical outcome cannot be used because each patient is treated in only one way and because a poor outcome in a gravely ill patient cannot necessarily be blamed on the treatment chosen. Second, although MYCIN performed at or near expert level in almost all cases, the evaluating experts in one study had serious reservations about the clinical utility of the program. It is difficult to assess how much of this opinion is due to actual inadequacies in system knowledge and design and how much is related to inherent bias (and according to Shortliffe, physicians are notoriously biased) against any computer-based consultation aid. Finally, MYCIN did not do well with patients who had serious infections simultaneously present at sites in the body about which the program has been given no rules. A useful antimicrobial consultation system must know about a broad range of infectious diseases. Thus, the program is not ready for clinical implementation, and we cannot test its usefulness or whether or not it is cost-effective.

5. The PROSPECTOR Consultant System

PROSPECTOR is a computer consultant system intended to aid geologists in evaluating the favorability of an exploration site or region for occurrences of ore deposit of particular types. Knowledge about a particular type of ore deposit is encoded in a computational model representing observable geological features and the relative significance thereof. The system is used by geologists exploring the hard-rock mineral deposit. However, it is intended to use the system in on-site explorations--especially for the petroleum industry--by the use of Lisp machines. In this case the users may well have less expertise than the designers.

A Lisp-based model description language is used and may well make use of the "subjective Bayesian methods" described in the following chapter. Since there are serious intentions to field test the program for commercial purposes, one may assume that it has been found useful. The following is taken from Ref. 24.

> The system is still in the process of being developed. PROSPECTOR is intended to emulate the reasoning process of an experienced exploration geologist in assessing a given prospect site or region for its likelihood of containing an ore deposit of the type represented by the model he or she designed. The term "model" refers to a body of knowledge about a particular domain of expertise that is encoded into the system and on which the system can act. The empirical knowledge contained in PROSPECTOR consists of a number of such specially encoded models of certain classes of ore deposits. The performance of PROSPECTOR depends on the number of models it contains, the types of deposits modelled, and the quality and completeness of each model. Because the PROSPECTOR program is primarily a research project, its coverage is still incomplete. It currently contains five prospect-scale models, one region-scale model, and one drilling site selection model. These models were selected for a variety of reasons, including their economic significance, the extent to which they are well understood scientifically, the availability of expert geologists who could collaborate with us in the model development, and the new research issues that their implementation would raise.

From the same reference, the following may be of value in assessing the effectiveness of the present system.

> In PROSPECTOR's normal interactive consultation mode, the user is assumed to have obtained some promising field data and is assumed to desire assistance in evaluating the prospect. This, the user begins by providing the program with a list of rocks and minerals observed, and by inputting other observations expressed in simple English sentences. The program matches these data against its models, requests additional information (in definitive questions) of potential value for arriving at more definite conclusions, and provides a summary of the findings (expressing its conclusions to the user on a -5 to 5 certainty scale that the user also employs to express his certainty about evidence requested by the system). The user can ask at any time for an elaboration of the intent of a question, or for the geological rationale for including a question in the model, or for an on-going trace of the effects of his answers on PROSPECTOR's conclusions. The intent is to provide the user with many of the services that could be provided by giving him telephone access to a panel of senior economic geologists, each an authority on a particular class of ore deposits.
>
> Since the PROSPECTOR system has only been used experimentally thus far, the only way to gage how well it works is through performance evaluation techniques which assess the extent to which the performance of the model reflects faithfully the intent of the model designer. The results of these evaluations specify portions of the model that might benefit from "fine tuning," and establish priorities for such revisions. How well any given model will perform is, of course, dependent upon the model designer, and upon the size,

the complexity, and how authoritative and up-to-date a model he has designed. (This last aspect, by the way, points to the need, especially in the use of scientific models, for constant update). The models currently in use have, as has been pointed out, been specially chosen, and thus might not prove an adequate measure of PROSPECTOR's effectiveness. It should, however, be pointed out that each model is encoded as a separate data structure, independent of the PROSPECTOR system per se. Thus, the PROSPECTOR program should not be confused with its models. (These models can always be tested and improved. The program can be run, and accidental blunders or bugs can be corrected. In addition, the program can produce a questionnaire for the model that is useful in gathering data for subsequent testing and revision). Rather, PROSPECTOR should be thought of as a general mechanism for delivering relevant expert information about ore deposits to a user who can supply it with data about a particular prospect or region. One general point about model design--some care must be exercised to avoid "overfitting" the model to the data. In general, the goal is to produce a model that can discriminate different types of deposits without losing the ability to generalize, so as to allow for the variations one would expect in new situations. Achievement of that, however (as the authors point out), remains as much an art as a science as yet. More extensive performance evaluation techniques and results are reported in the above cited article. There are some advantages to the model design process. This process challenges the model designer to articulate, organize, and quantify his expertise without exception, the economic geologists who have designed PROSPECTOR models have reported the experience aided and sharpened their own thinking on the subject matter of the model. Most have remarked on PROSPECTOR's potential value as an educational tool--the models in this system contain explicit, detailed information synthesized from the literature and the experience of expert explorationists, together with explanatory text that can be obtained upon request. Furthermore, a typical consultation session with PROSPECTOR costs only about $10 at current commercial computer rates.

6. DENDRAL and META-DENDRAL

The Heuristic DENDRAL program is designed to help organic chemists determine the molecular structure of unknown compounds. Because of the wealth of data that can be acquired from a mass spectrometer, an automatic rule-formation program, META-DENDRAL, was developed. This acquires new rules in two ways: through interaction with an expert and through automatic hypothesis formation. The reason for adopting this strategy is (Ref. 25) that:

> Because of the difficulty of extracting domain-specific rules from experts for use by DENDRAL, a more efficient means of transferring knowledge into the system was sought. One was the use of interactive knowledge transfer programs; the other was the use of automatic theory formation programs. In some areas of chemistry, there are no experts with enough specific knowledge to make a high-performance problem-solving program--besides which, it is a time-consuming process. In the absence of this interactive transfer of knowledge, an effort to build an automatic rule formation program (called META-DENDRAL) was initiated. The DENDRAL programs are structured to read their task-specific knowledge from tables of production rules and execute the rules in new situations under rather elaborate control structures. The META-DENDRAL

programs (working with chemists to determine the dependence of mass spectrometric fragmentation on substructural features) have been constructed to aid in building the knowledge data base (i.e., the tables of rules).

DENDRAL is largely coded in INTERLISP. Parts of the hypothesis generator, CONGEN, are coded in SAIL and Fortran. At the time of the original articles on this system, an attempt was being made to rewrite the code in lower level languages to make it more efficient and to allow it to run on a greater variety of machines. It is assumed that this attempt has been abandoned.

The following describes the state of use of the system:

CONGEN has attracted a moderately large following of chemists who consult it for help with structure elucidation problems. INTSUM (the planning part of the META-DENDRAL program which collects and summarizes evidential support), too, is used occasionally by persons collecting and codifying a large number of mass spectra, With these noted exceptions, the DENDRAL and META-DENDRAL programs are not used outside the Stanford University Community and thus they represent only a successful demonstration of scientific capability.

7. Computer Configurers

The preceding sections will have indicated the point made at the outset of this chapter, that expert systems have yet to be justified commercially. There is one outstanding success in this development--that of computer configurers.

Rl is a system that is in day-to-day use for configuring VAX-11/780 computers. It constructs from customer specifications and interaction with a representative to specify both the physical connections and the spatial arrangement of a computer system (Ref. 26). As it is currently implemented, R2 appears to contain about 3,000-4,000 rules and is implemented in OPS-4, a production system language whose successor, OPS-5, is discussed below. A major problem in the implementation of the system appears to have been tuning the language to adequate efficiency.

The success of this program requires no further comment since its use is widespread. The user interface is a straightforward question-and-answer dialog. It is noteworthy that other manufacturers have achieved similar successes: in particular, there is a system at Burroughs under development that is implemented in Prolog.

8. Data Bases and Expert Systems

All the systems cited above have a purpose-built data base or knowledge base. For the most part, these have been constructed by individuals with an extensive knowledge of the subject matter. It is possible that components of the Rl rule base could have been generated automatically from the appropriate inventory files, but this is unlikely to account for a substantial fraction of the knowledge base.

In order to find out whether existing data bases could be used at all in expert systems, we should identify the kinds of problems for which expert systems might be developed and the nature of the relevant data bases. We should also investigate the methods by which factual data can be used in expert systems.

The use of expert systems has been widely discussed with respect to Command and Control, but in assessing their applicability we should distinguish between decision support systems and expert systems. The former generally comprise data presentation and analysis techniques and predictive modelling. The latter usually involves some deductive reasoning system that attempts some form of problem-solving. Decision support, especiallv related to data presentation, has been extremely useful in Command and Control, but it appears that there are yet no adequate prototypes of relevant expert systems. There are two possible reasons for this. One is that the knowledge base is too large and heterogeneous to be susceptible to existing techniques; note that all the examples cited above had reasonably uniform knowledge bases. The other is that it may be difficult to produce a system whose response time and reliability is adequate.

One of the most demanding functions in Command and Control is monitoring for occurrence of certain events which are indicated by relatively complicated logical inferences. Although it is probably too soon to expect useful expert systems to subsume the whole function of Command and Control, it may well be that independent, automated agents could be constructed to perform some of the more sophisticated and time-consuming monitoring tasks. There are a few proposals for real-time expert systems and it could be that, with the appropriate user interface, some of the "expert" monitoring functions could be automated.

9. The Use of Data Bases

Expert systems are usually rule-based, the individual rules describing the deductions that the system is allowed to make. Conventional data bases can be viewed as a collection of facts, but no rules. At first sight, the possibility of using a conventional data base in an expert system may appear remote. However, there is a sense in which a fact can be regarded as a simple rule and this has been shown in a number of deductive systems of which the best known is the Logic programming language Prolog.

A Prolog data base consists of a collection of facts and rules. The stock example is that of a family tree:

```
father(john, paul).
mother(mary, paul).
mother(ann, mary).
father(henry, mary).
mother(ann, joe).
father(henry, joe).
father(joe, bill).
mother(ellen, bill).
```

```
sibling(X,Y) :- mother(M,X), mother(M,Y),
                father(F,X), father(F,Y)
grandfather(X,Y) :- father(X,Z), father(Z,Y).
grandfather(X,Y) :- father(X,Z), mother(Z,Y).
parent(X,Y) :- mother(X,Y).
parent(X,Y) :- father(X,Y).
child(X,Y) :- parent(Y,X).
ancestor(X,X).
ancestor(X,Y) :- parent (X,Z), ancestor(Z,Y).
```

The facts in this data base are the statement such as "father(john,paul)", which is a representation of the fact that "john is the father of paul". The facts are special cases of rules, that are of the form <consequent> :- <antecedent>. That is, the rule

```
sibling(X,Y) :- mother(M,X), mother(M,Y),
                father(F,X), father(F,Y).
```

reads "X is a sibling of Y if M is mother of X and M is mother of Y and F is father of X and F is father of Y". In response to typing in the

```
? - question sibling(X,joe).
```

the Prolog interpreter will attempt to match variables (the variables are in upper case) in order to prove the assertion that "There is an X for which X is the sibling of joe". In this case, the proof succeeds with the X in sibling(X,joe) matched to mary. Note that the variables are words starting with an upper-case character.

A more complicated example is the ancestor relationship. The two rules assert first that X is an ancestor of X, i.e., anyone is an ancestor of himself. The second asserts that X is an ancestor of Z if X if parent of Y and Y is ancestor of Z. In response to the question ancestor(X,paul), the Prolog interpreter will find, on demand, all those instances of X which make the predicate true. A more general question is ancestor (X,Y) in which Prolog will find all instances of X and Y that make the predicate true.

Prolog has a syntax for lists built into it, thus list processing predicates may be created in a manner similar to the way they are constructed in Lisp. To append a list X to a list Y, for example:

```
append([],X,X).
append([A|X],Y[A|Z]) :- append(X,Y,Z).
```

This reads that the empty list [] appended to X is X and a list whose head is A and whose tail is X ([A|X]) appended to Y is the list whose head is A and whose tail is Z ([A|Z],) where Z is the result of appending X to Y. Note that this will not only instantiate X to [1,2,3,4] in append([1,2],[3,4],X) but it will also make the successive instantiations:

X	Y
[]	[1,2,3,4]
[1]	[2,3,4]
[1,2]	[3,4]
[1,2,3]	[4]
[1,2,3,4]	[]

in response to append(X,Y,[1,2,3,4]). Conventional arithmetic is implemented through the use of the pseudo-predicate; thus, N is 3+4 will bind N to 7. However, 7 is X+Y will fail, i.e., it will not produce bindings for X and Y.

Consider, for example, how the function factorial might be implemented in Prolog.

```
factorial(0,1).
factorial(N,M) :- Y is N-1, factorial(Y,Z), M is N*Y.
```

Given, for example, factorial(4,X), X will first be correctly bound to 24. However, the attempt to find further solutions will not stop. The initial solution successively matched factorial (4,_), factorial(3,_), factorial(2,_), factorial(1,_), and factorial(0,_). The last of these would have succeeded with the use of the first clause. In looking for further solutions, the backtracking mechanism proceeds past this match and computes factorial(-1,_), factorial(-2,_), etc. Thus, asking for another match will cause an endless loop. There are two solutions to this problem: one is to add a condition in the right-hand side of the second clause that N > > 1. The other is to add a "cut" to the first clause. This indicates that once the first predicate has matched, no further backtracking can take place past this match. It should also be noted that any other arrangement of the "conditions" on the left-hand side of the second clause will cause the factorial predicate to fail in general.

It should be apparent that the collection of facts in a Prolog data base can be thought of as a relational data base. The difference is that the columns of the relation are coded positionally rather than by tags. Thus, the set of assertions

```
father(john, paul).
father(henry, mary).
father(henry, joe).
father(joe, bill).
```

would be held in a relation FATHER(PARENT, CHILD) in which the components are defined by labels (PARENT and CHILD) whereas in Prolog they are held positionally.

For data base work this is more than a minor inconvenience because relations may, and often do, have tens or even hundreds of columns. It is awkward to remember these by position. Using labels in a Prolog-like language has recently become a matter of considerable research interest. There is every expectation that more sophisticated data models can be integrated with problem-solving languages and other tools for the construction of expert systems.

For the time being, however, the best methods of integrating existing data bases with any of these problem-solving systems is to provide a clean, interactive, relational interface. Any long responses from the data base will almost certainly mean that the data base will have to be converted to the appropriate rule base before it can be used.

REFERENCES

1. CCITT Standard X.200, Reference Model of Open Systems Interconnection for CCITT Applications.

2. "Specification for the Interconnection of a Host and an IMP," Report No. 1822, Bolt, Beranek and Newman Inc., Cambridge, MA, revision of December 1981.

3. CCITT Recommendation X.25, "Interface Between Data Terminal Equipment (DTE) and Data Circuit Terminating Equipment (DCE) for Terminals Operating in the Packet Mode on Public Data Networks," International Telegraph and Telephone Consultative Committee Yellow Book, Vol. VIII.2, Geneva, 1981.

4. "Defense Data Network X.25 Host Interface Specification," BBN Report No. 5476, Cambridge, MA, December 1983.

5. "Defense Data Network Subscriber Interface Guide," Defense Communications Agency, Washington, DC, July 1983.

6. "Internet Protocol Transition Workbook," SRI International, Menlo Park, CA, March 1982.

7. "Internet Protocol Implementation Guide," SRI International, Menlo Park, CA, August 1982.

8. "DDN Subnet Access Protocol Interoperability Study," BBN Report No. 5360, Cambridge, MA, October 1983.

9. Blankertz, W.H., and D.A. Gombert, "WIS Local Area Network Issues," Mitre Report MTR-82W00123, July 1982.

10. Shirey, R.W., "FY81 Final Report: Cable Bus Applications in Command Centers, Security Issues," Mitre Report MTR-81W00248-02, February 1982.

11. Ware, W. "Security Controls for Computer Systems, Report of the Defense Science Board Task Force on Computer Security," Rand Corporation R-609-1, originally published February 1970, reissued October 1979.

12. Chorafas, D.N., "Designing and Implementing Local Area Networks," McGraw-Hill, 1984.

13. Sidhu, D.P., and M. Gasser, "Design for a Multilevel Secure Local Area Network," Mitre Report MTR-8702, March 1982.

14. Institute for Defense Analyses, "C^3I Data Base and Networking Analysis," IDA Paper P-1637, T.C. Bartee and O.P. Buneman, April 1982.

15. Institute for Defense Analyses, "C^3I Local Area Networks--An Assessment," IDA Paper P-1715, T.C. Bartee and O.P. Buneman, June 1983.

16. Van de Riet, R.P., et al. "High Level Language Features for Improving the Efficiency of a Data Base System," ACM Trans. on Data Base Systems 6 (3): 1981.

17. Rowe, L.A., Reference Manual for the Programming Language RIGEL, Technical Report, University of California at Berkeley, Department of Electrical Engineering, 1980.

18. Buneman, O.P., J.B. Hirchberg and D.J. Root, "A Codasyl Interface for Pascal and Ada," in Proceedings, British National Conference on Data Bases, 1982.

19. Atkinson, M.P., et al. "An Approach to Persistent Programming," Computer Journal 26 (4), November 1983.

20. Shipman, D.W., "The Functional Data Model and the Data Language DAPLEX," ACM Trans. on Data Base Systems, 6 (1); 140-173, March 1981.

21. Schmidt, J.W., "Some High Level Language Constructs for Data of Type Relation," ACM Trans. on Data Base Systems, 2 (3): 247-281, September 1977.

22. Smith, J.M., S. Fox and T. Landers, ADAPLEX: Rationale and Reference Manual, Second Edition, Computer Corporation of America, Four Cambridge Center, Cambridge, MA, 1983.

23. Pople, H.E., Jr., "The Formation of Composite Hypothesis in Diagnostic Problem Solving/An Exercise in Synthetic Reasoning," in Proc. IJACI-5, pp. 1030-1037, IJACI, 1977.

24. Duda, R.O., and J.S. Gaschnig, "Knowledge-Based Expert Systems of Age," Byte 6 (9): 238-81, 1981.

25. Feigenbaum, E.A., et al. "On Generality and Problem Solving--A Case Study Involving the DENDRAL Program," in D. Michie and B. Meltzer (editors), Machine Intelligence 6, Edinburgh University Press, 1971.

26. McDermott, J., R1: A Rule-Based Configurer of Computer Systems, Technical Report CMU-CS-80-119, Carnegie Mellon University, 1980.